Как написать свой индекс в Tarantool

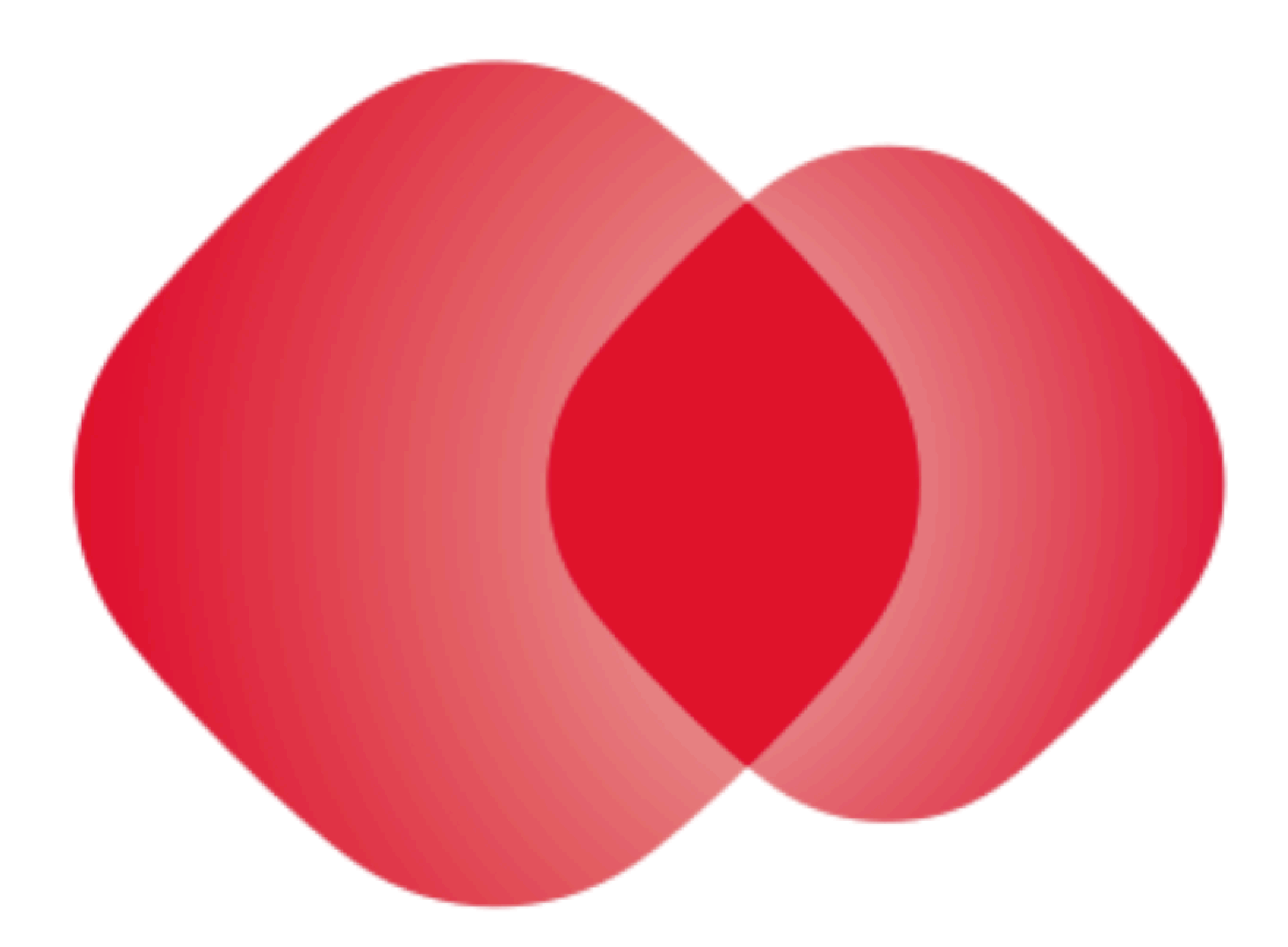
Бабин Олег



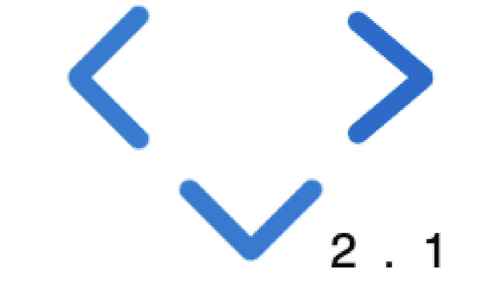


Tarantool

- Сервер приложений
- База данных
  - memtx — хранение в памяти
  - vinyl — хранение на диске



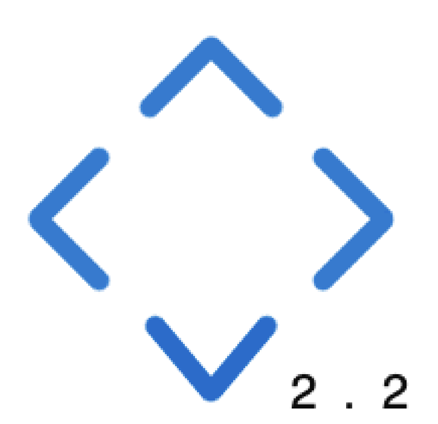




Первый запуск

```
box.cfg{}
box.schema.space.create('my_space')
box.space.my_space:create_index('primary',
        {parts = {{field = 1, type = 'unsigned'}}})
box.space.my_space:replace({1, 'hello'})
box.space.my_space:replace({2, 'world'})
```





Первый запуск

```
box.cfg{}
box.schema.space.create('my_space')
box.space.my_space:create_index('primary',
        {parts = {{field = 1, type = 'unsigned'}}})
box.space.my_space:replace({1, 'hello'})
box.space.my_space:replace({2, 'world'})
```

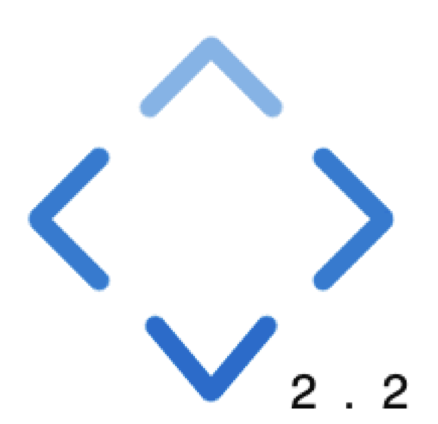




Первый запуск

```
box.cfg{}
box.schema.space.create('my_space')
box.space.my_space:create_index('primary',
            {parts = {{field = 1, type = 'unsigned'}}})
box.space.my_space:replace({1, 'hello'})
box.space.my_space:replace({2, 'world'})
```

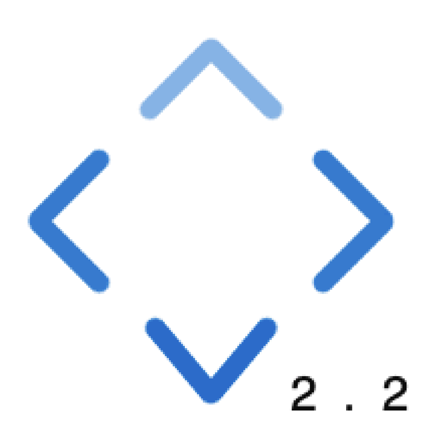




Первый запуск

```
box.cfg{}
box.schema.space.create('my_space')
box.space.my_space:create_index('primary',
        {parts = {{field = 1, type = 'unsigned'}}})
box.space.my_space:replace({1, 'hello'})
box.space.my_space:replace({2, 'world'})
```

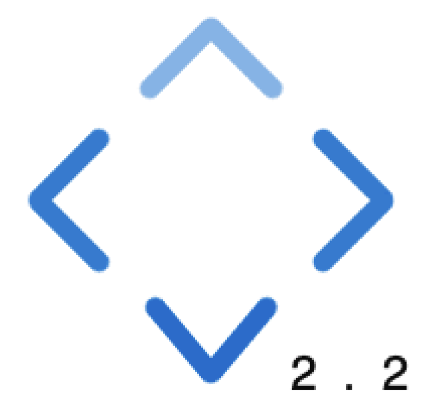




Первый запуск

```lua
box.cfg{}
box.schema.space.create('my_space')
box.space.my_space:create_index('primary',
        {parts = {{field = 1, type = 'unsigned'}}})
box.space.my_space:replace({1, 'hello'})
box.space.my_space:replace({2, 'world'})
```

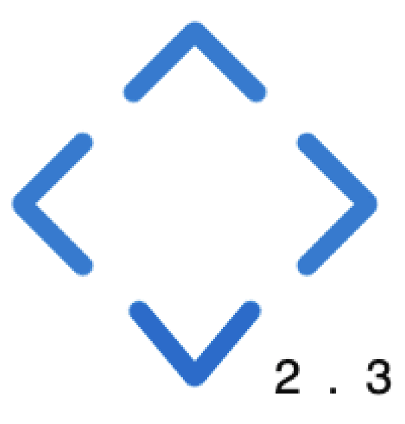




Выборка данных

```
tarantool> box.space.my_space:select()
---
- - [1, 'hello']
  - [2, 'world']
...
tarantool> box.space.my_space:select({1},
         >     {iterator = box.index.GT})
---
- - [2, 'world']
...
```

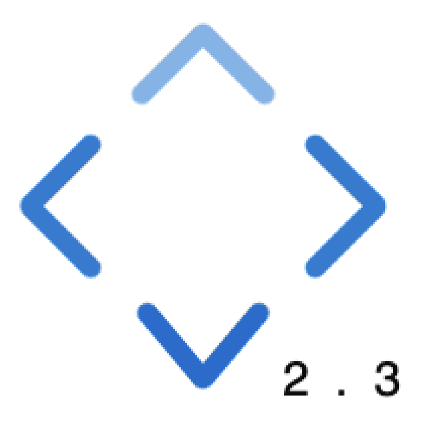




Выборка данных

```
tarantool> box.space.my_space:select()
---
- - [1, 'hello']
  - [2, 'world']
...
tarantool> box.space.my_space:select({1},
         >     {iterator = box.index.GT})
---
- - [2, 'world']
...
```

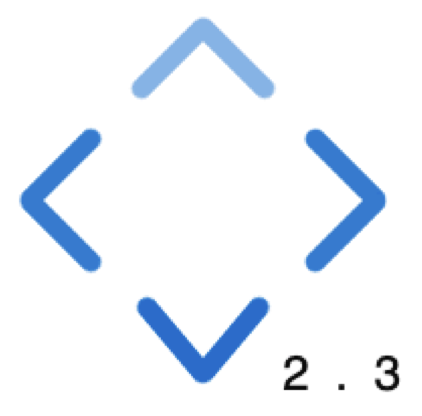




Выборка данных

```
tarantool> box.space.my_space:select()
---
- - [1, 'hello']
  - [2, 'world']
...
tarantool> box.space.my_space:select({1},
         >     {iterator = box.index.GT})
---
- - [2, 'world']
...
```

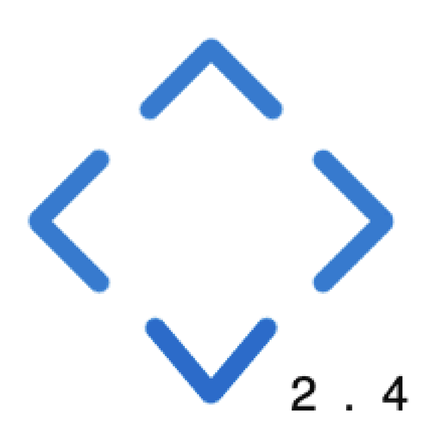




Вторичные индексы

```
space = box.space.my_space
space:create_index('secondary',
            {parts = {{field = 2, type = 'string'}}})
tarantool> space.index.secondary:select({'world'})
---
- - [2, 'world']
...
```

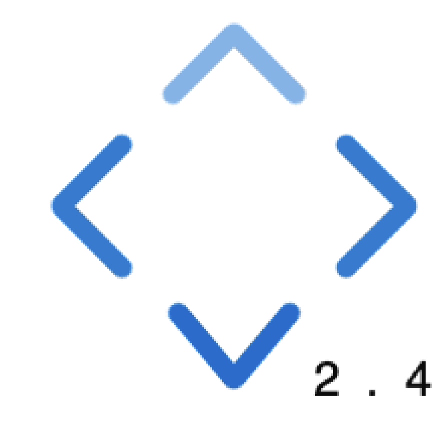




Вторичные индексы

```
space = box.space.my_space
space:create_index('secondary',
        {parts = {{field = 2, type = 'string'}}})
tarantool> space.index.secondary:select({'world'})
---
- - [2, 'world']
...
```

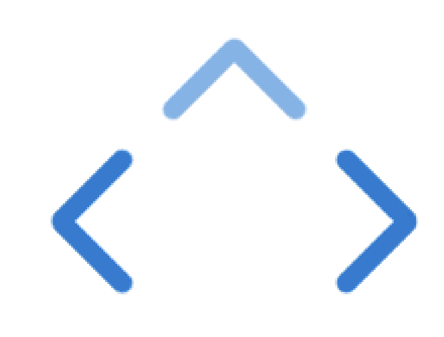




Вторичные индексы

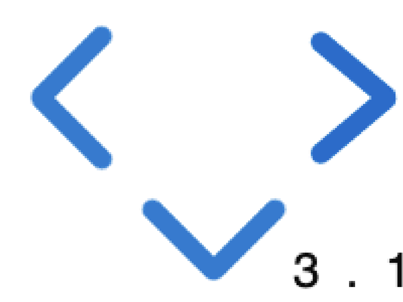
```
space = box.space.my_space
space:create_index('secondary',
          {parts = {{field = 2, type = 'string'}}})
tarantool> space.index.secondary:select({'world'})
---
- - [2, 'world']
...
```





Многомерные данные

HighLoad++
Весна 2021



Смартфон Xiaomi Redmi 9 4/64GB (NFC)

4.6 160 отзывов

долгое время работы, объем памяти

Android 10
поддержка двух SIM-карт
экран 6.53", разрешение 2340x1080
4 камеры: широкоугольная (13 МП), сверхширокоугольная, макро, датчик глубины
процессор MediaTek Helio G80

6 606 человек купили этот товар

1 из 6 фото

11 599 ₽
от 437 ₽/мес
75 предложений от 11 528 ₽

Четкий экран, яркости мне хватает. Наклеенная защитная пленка "с завода"....
Андрей В.

Смартфон Apple iPhone 11 128GB

4.7 319 отзывов

качество фотографий, мощный процессор

iOS 13
поддержка двух SIM-карт (nano SIM+eSIM)
экран 6.1", разрешение 1792x828
двойная камера: 12 МП, 12 МП
процессор Apple A13 Bionic

8 922 человека купили этот товар

1 из 6 фото

55 390 ₽
327 предложений от 55 390 ₽

Уникальный аппарат, который уверенно смотрит в будущее, да и сам - уже в...
Ольга К.

Смартфон Samsung Galaxy A71 6/128GB

4.6 273 отзыва

экран, объем памяти

Android 10
поддержка двух SIM-карт
экран 6.7", разрешение 2400x1080
4 камеры: 64 МП, 12 МП, 5 МП, 5 МП
процессор Qualcomm Snapdragon 730

6 011 человек купили этот товар

1 из 6 фото

22 650 ₽
140 предложений от 22 450 ₽

Большой, красочный экран Время автономной работы Быстрая зарядка...
Максим Т.

Смартфон Xiaomi Redmi 9A

4.5 80 отзывов

~~7 990 ₽~~ -7%
7 450 ₽
92 предложения от 7 339 ₽

Производитель
- Apple
- BQ
- HONOR
- HUAWEI
- Nokia
- OnePlus
- OPPO
- Philips
- realme
- Samsung
- Xiaomi
- ZTE

Показать всё

- [x] **В продаже**

Тип
- кнопочный телефон
- смартфон

Платформа
- Android
- iOS

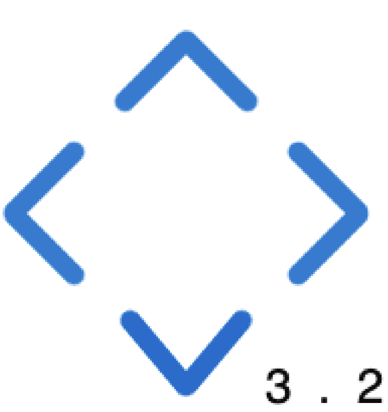
- **Игровой**

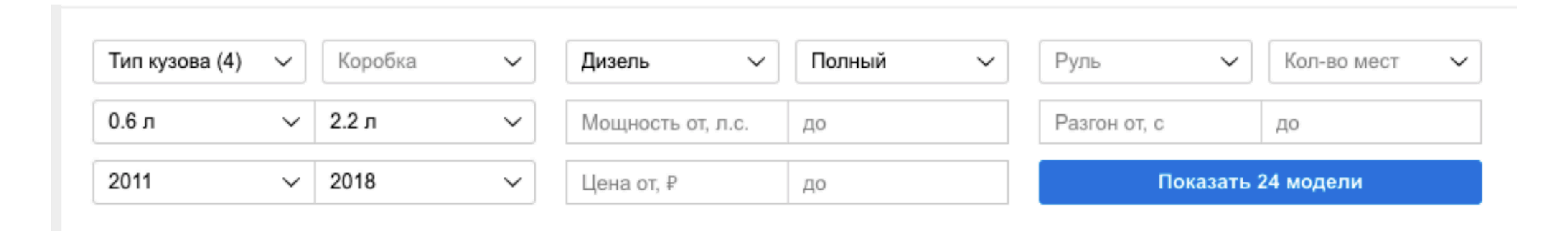
Диагональ экрана (точно), "

от 0.66 | до 7.6

Диагональ экрана
- до 3.4"
- 3.5"-4.9"
- 5.0"-5.4"
- 5.5"-5.9"
- 6.0"-6.4"
- 6.5" и больше

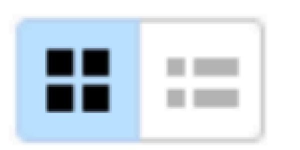


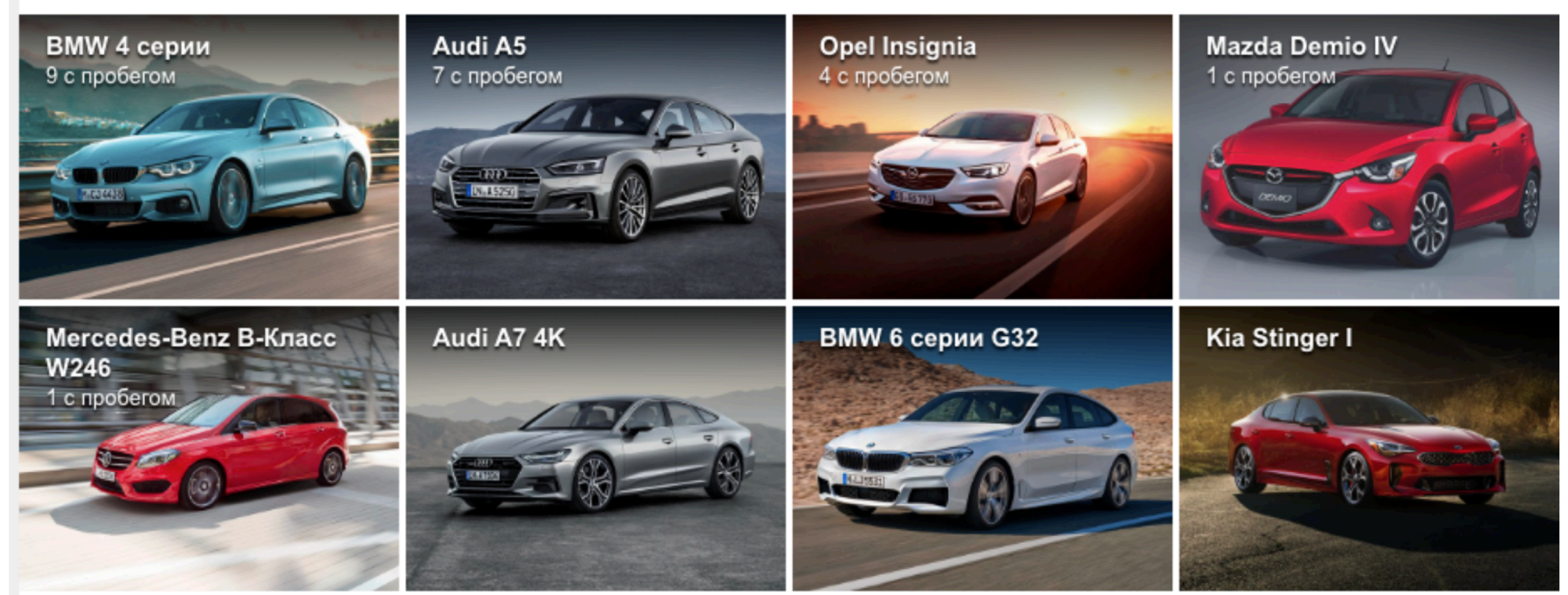




Тип кузова (4)
Коробка
Дизель
Полный
Руль
Кол-во мест
0.6 л
2.2 л
Мощность от, л.с.
до
Разгон от, с
до
2011
2018
Цена от, ₽
до
Показать 24 модели

Популярные





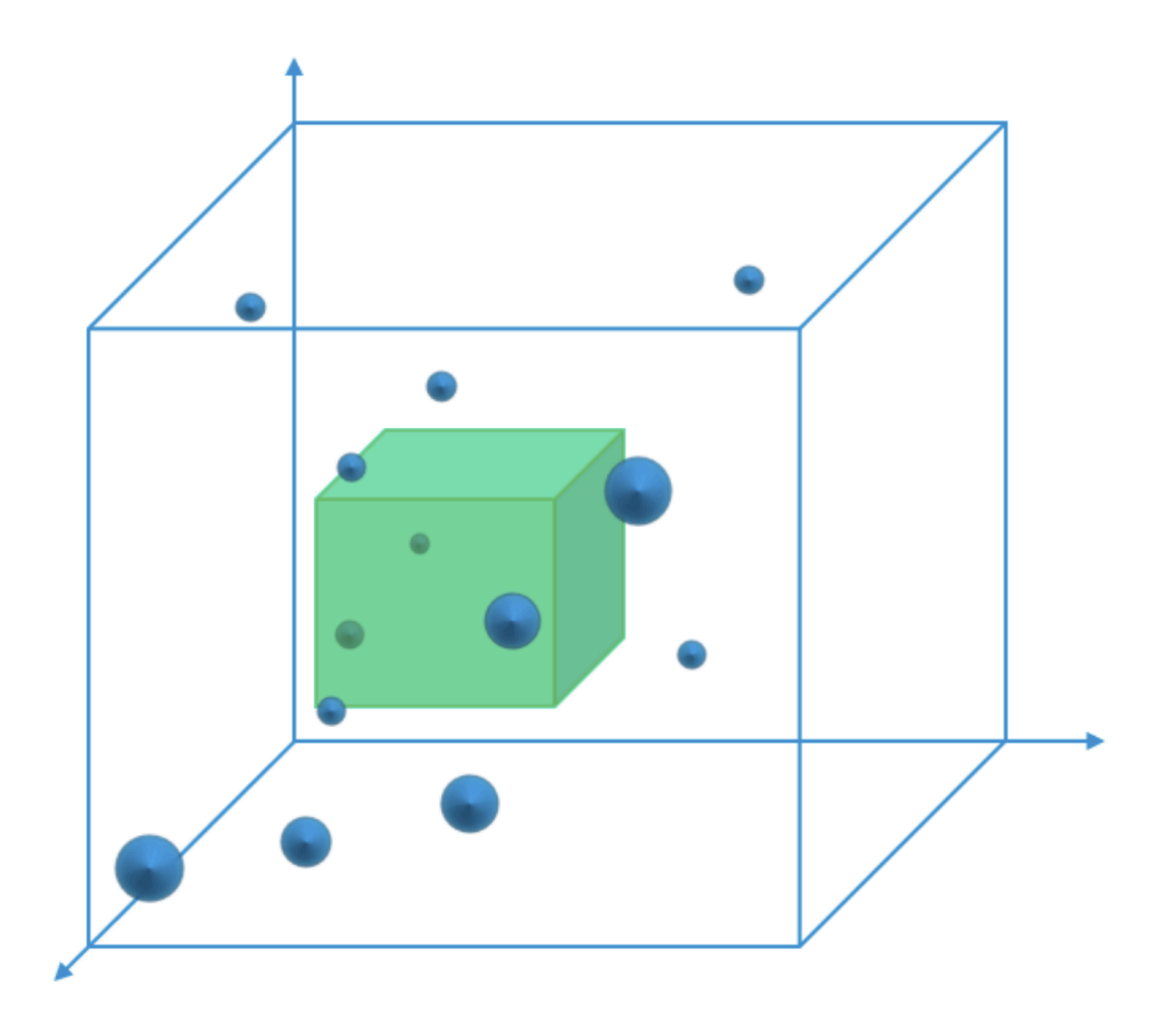
BMW 4 серии
9 с пробегом
Audi A5
7 с пробегом
Opel Insignia
4 с пробегом
Mazda Demio IV
1 с пробегом
Mercedes-Benz B-Класс W246
1 с пробегом
Audi A7 4K
BMW 6 серии G32
Kia Stinger I



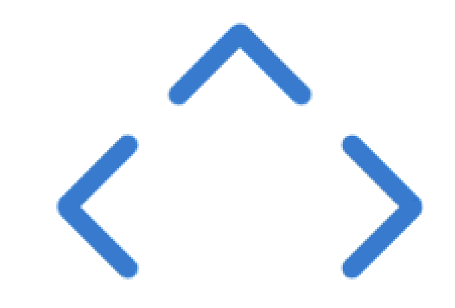
HighLoad++
Весна 2021



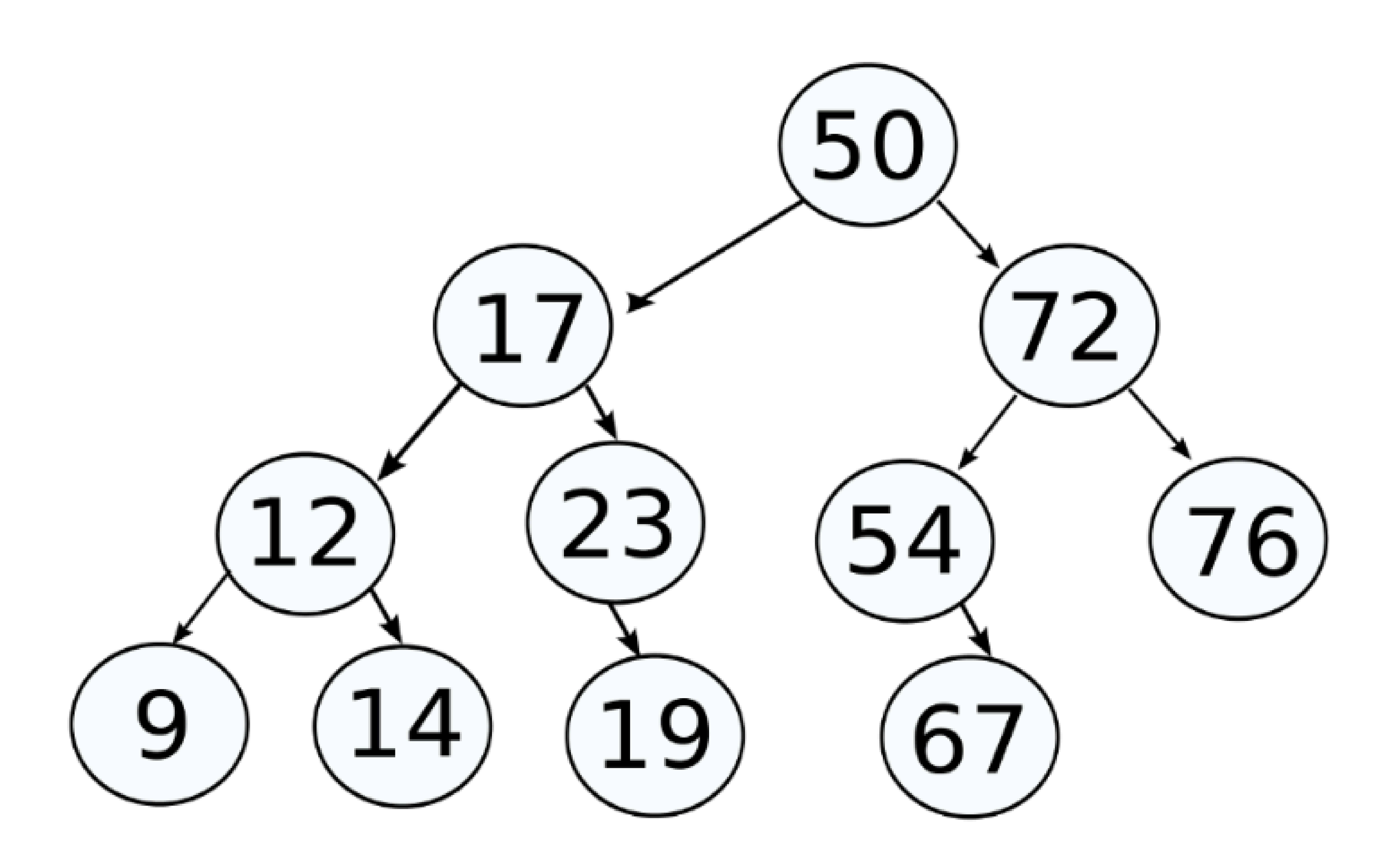
Многокритериальный поиск

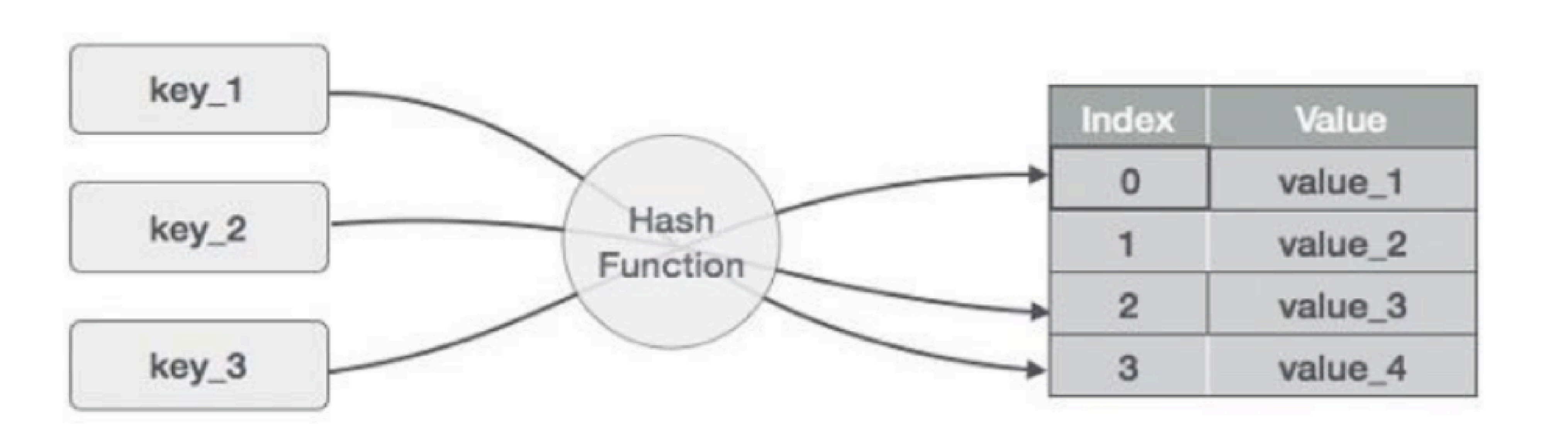






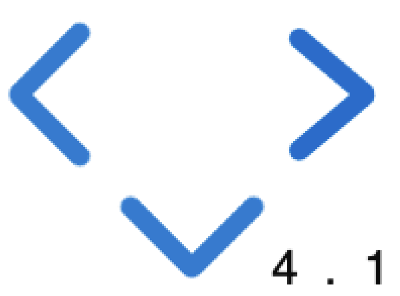
Индексы


50
17
72
12
23
54
76
9
14
19
67


key_1
key_2
key_3
Hash Function
Index
Value
0
value_1
1
value_2
2
value_3
3
value_4


3
Next
4
Next
5
Next
6
Next
null


HL
HighLoad++
Весна 2021



B-Tree

3 5

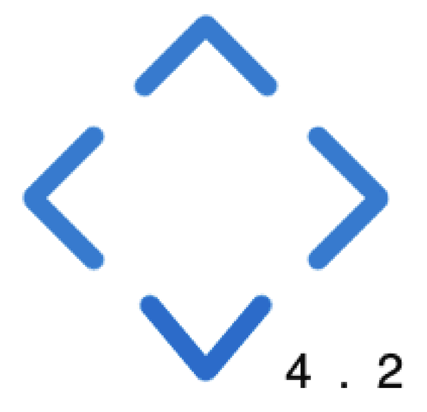
1 2

3 4

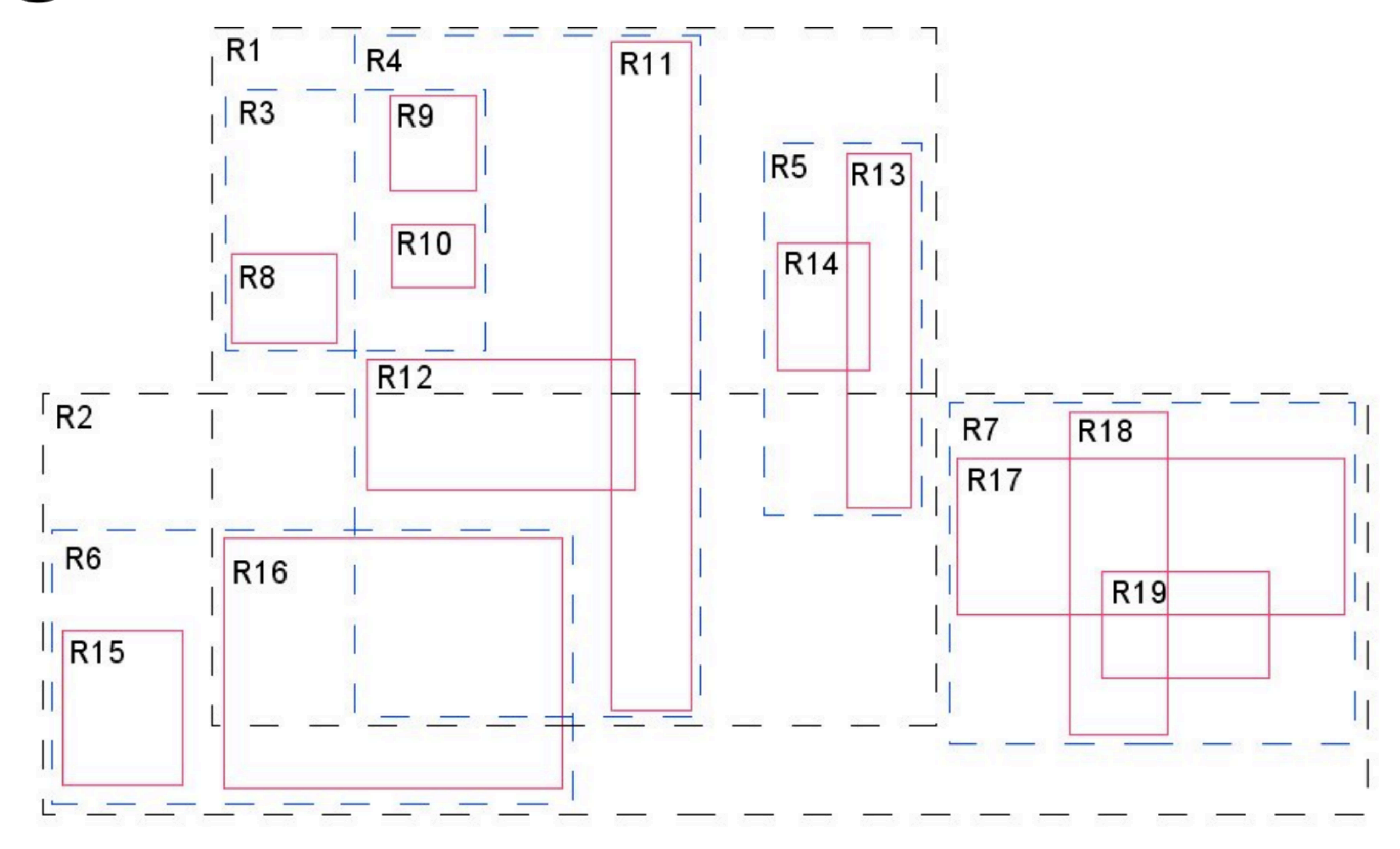
5 6 7

d_1 d_2 d_3 d_4 d_5 d_6 d_7

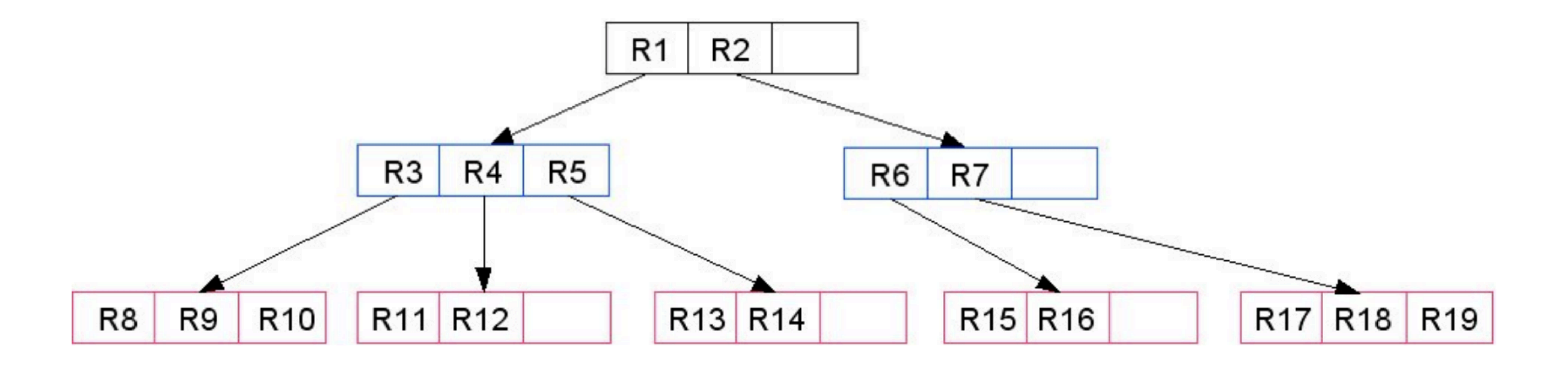




R-Tree



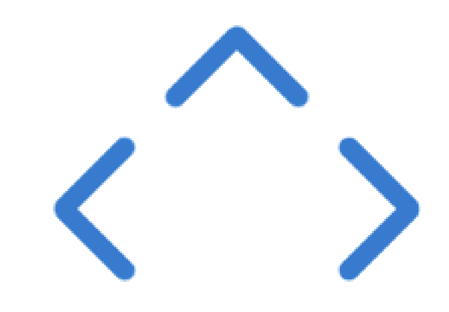
R1
R4
R11
R3
R9
R5
R13
R10
R8
R14
R12
R2
R7
R18
R17
R6
R16
R19
R15



R1 R2
R3 R4 R5
R6 R7
R8 R9 R10
R11 R12
R13 R14
R15 R16
R17 R18 R19



HL
HighLoad++
Весна 2021

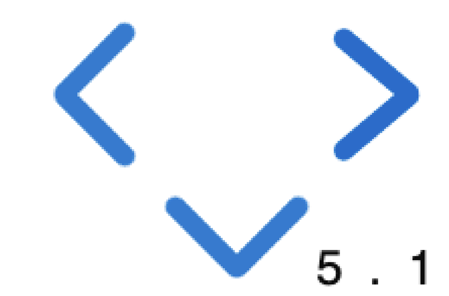


Как сейчас?

```lua
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})
for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, i, j})
    end
end
```

4				
3				
2				
1				
0	1	2	3	4

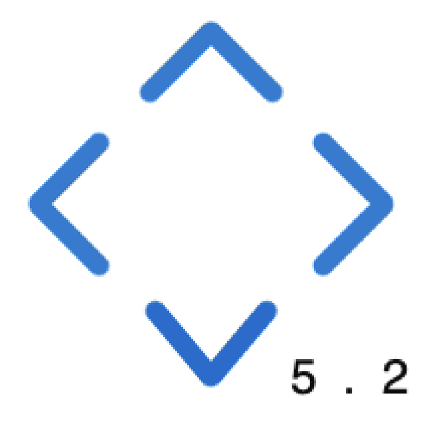




Наивный подход

```lua
result = {}
for _, tuple in space:pairs() do
    local x = tuple[2]
    local y = tuple[3]
    if (1 <= x and x <= 2) and (1 <= y and y <= 2) then
        table.insert(result, tuple)
    end
end
```





B-Tree

```
index = space:create_index('secondary', {
    type = 'tree',
    parts = {{2, 'unsigned'}, {3, 'unsigned'}}
})

result = {}
for _, tuple in index:pairs({1, 1}, 'GE') do
    local x = tuple[2]
    local y = tuple[3]

    if x >= 2 and y > 2 then
        break
    end

    if 1 <= y and y <= 2 then
        table.insert(result, tuple)
    end
end
```

B-Tree

```lua
index = space:create_index('secondary', {
    type = 'tree',
    parts = {{2, 'unsigned'}, {3, 'unsigned'}}
})

result = {}
for _, tuple in index:pairs({1, 1}, 'GE') do
    local x = tuple[2]
    local y = tuple[3]

    if x >= 2 and y > 2 then
        break
    end

    if 1 <= y and y <= 2 then
        table.insert(result, tuple)
    end
end
```

R-Tree

```
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})

for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, {i, j}})
    end
end

index = space:create_index('secondary', {
    type = 'rtree',
    parts = {{2, 'array'}},
    dimension = 2,
})

result = index:select({1, 1, 2, 2}, 'LE')
```

R-Tree

```
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})

for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, {i, j}})
    end
end

index = space:create_index('secondary', {
    type = 'rtree',
    parts = {{2, 'array'}},
    dimension = 2,
})

result = index:select({1, 1, 2, 2}, 'LE')
```

R-Tree

```
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})

for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, {i, j}})
    end
end

index = space:create_index('secondary', {
    type = 'rtree',
    parts = {{2, 'array'}},
    dimension = 2,
})

result = index:select({1, 1, 2, 2}, 'LE')
```

R-Tree

```
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})

for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, {i, j}})
    end
end

index = space:create_index('secondary', {
    type = 'rtree',
    parts = {{2, 'array'}},
    dimension = 2,
})

result = index:select({1, 1, 2, 2}, 'LE')
```

R-Tree

```
space:create_index('primary', {
    type = 'tree',
    parts = {{1, 'unsigned'}}
})

for i = 0, 4 do
    for j = 0, 4 do
        space:insert({i * 5 + j, {i, j}})
    end
end

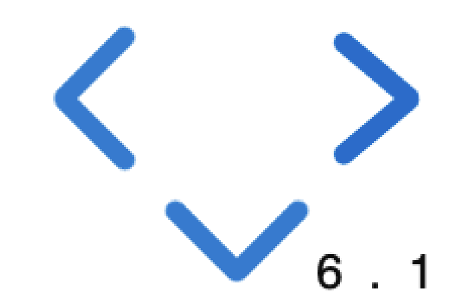
index = space:create_index('secondary', {
    type = 'rtree',
    parts = {{2, 'array'}},
    dimension = 2,
})

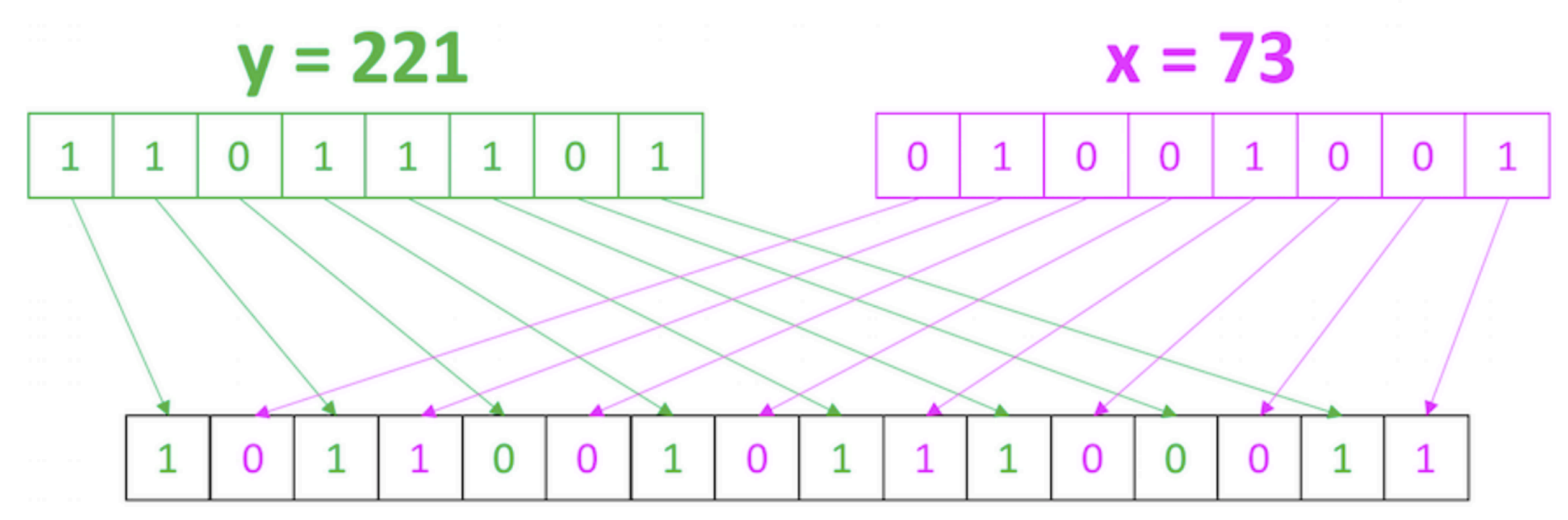
result = index:select({1, 1, 2, 2}, 'LE')
```

Кривая Z-порядка

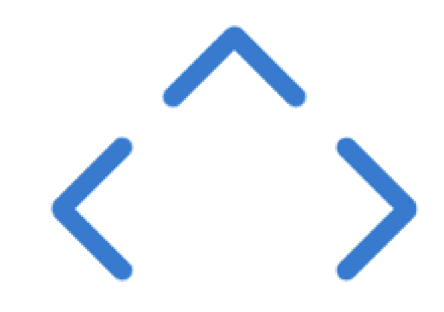
	x: 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111






y = 221
x = 73
1 1 0 1 1 1 0 1
0 1 0 0 1 0 0 1
1 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1
z-address = 45,795

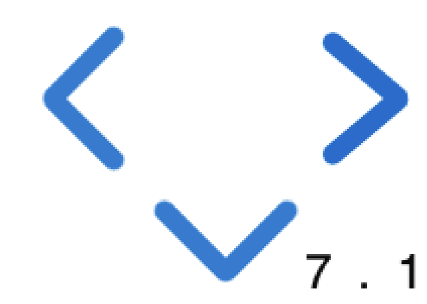

HL
HighLoad++
Весна 2021



Поиск

y: \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111





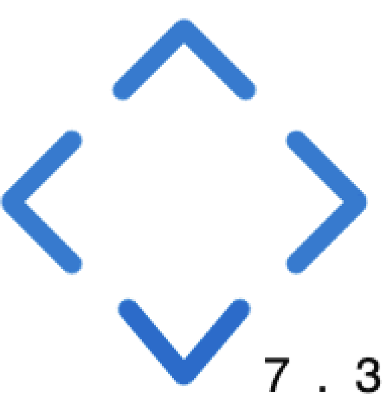
y \ x	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

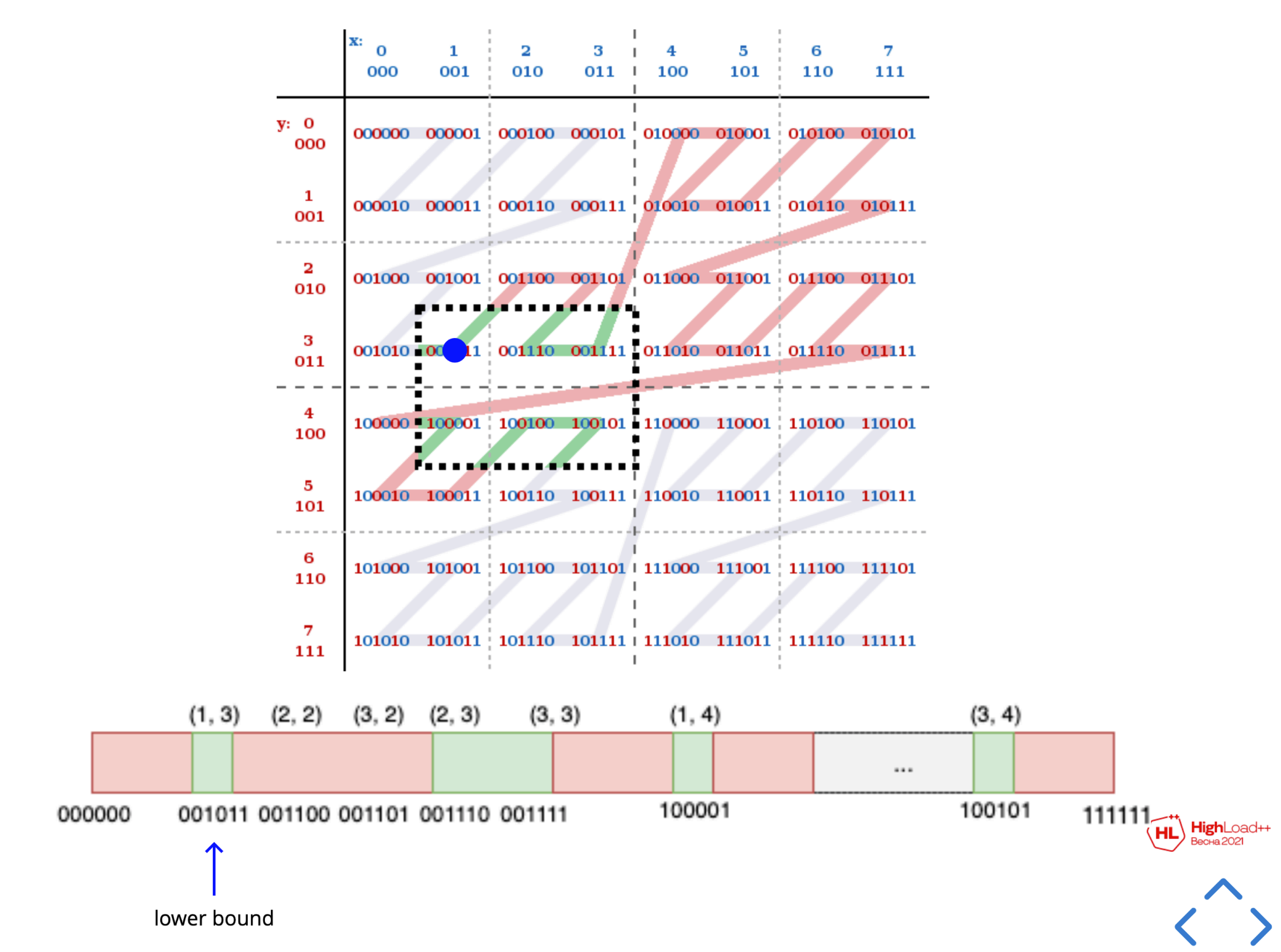


y \ x	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

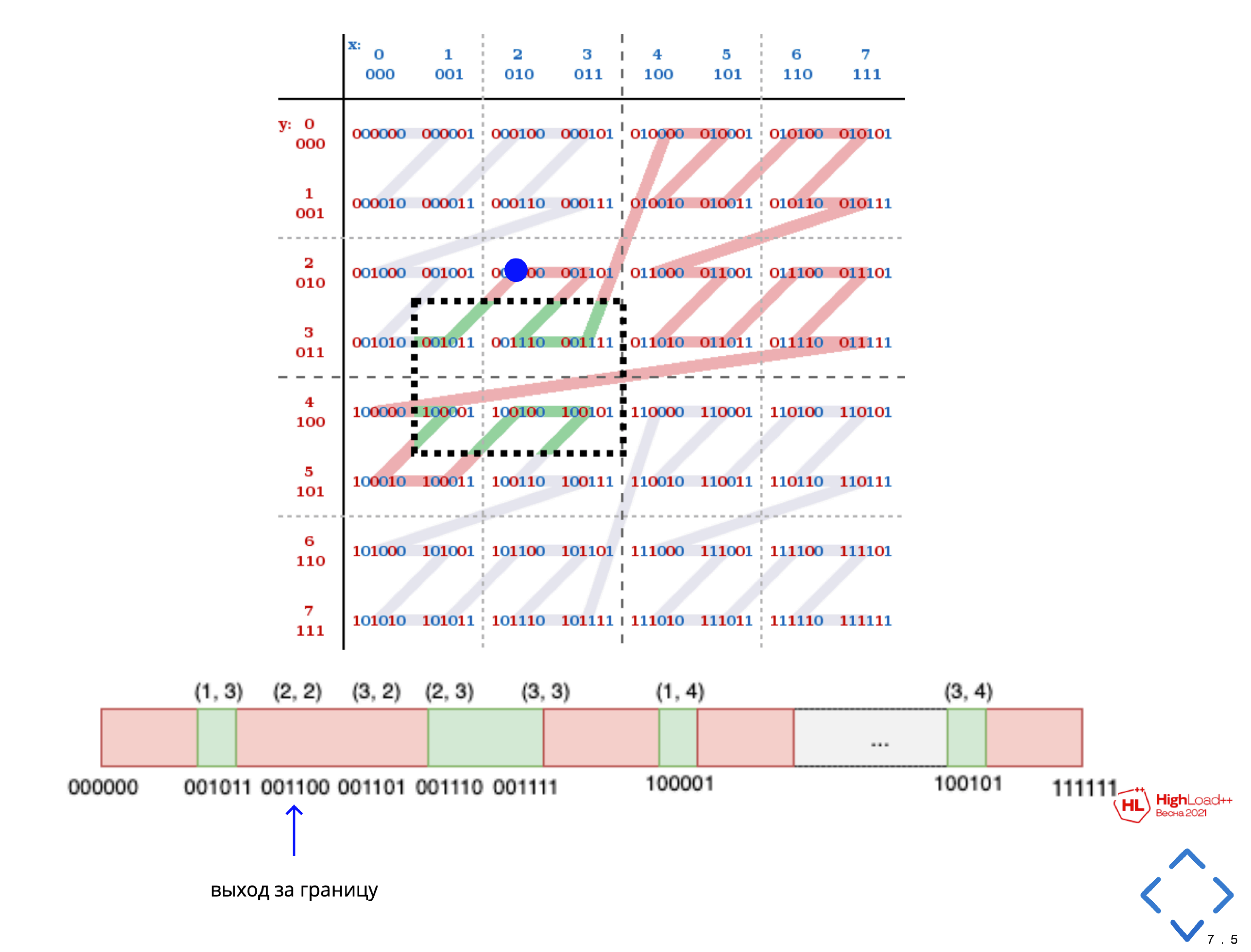
(1, 3) (2, 2) (3, 2) (2, 3) (3, 3) (1, 4) ... (3, 4)

000000 001011 001100 001101 001110 001111 100001 100101 111111

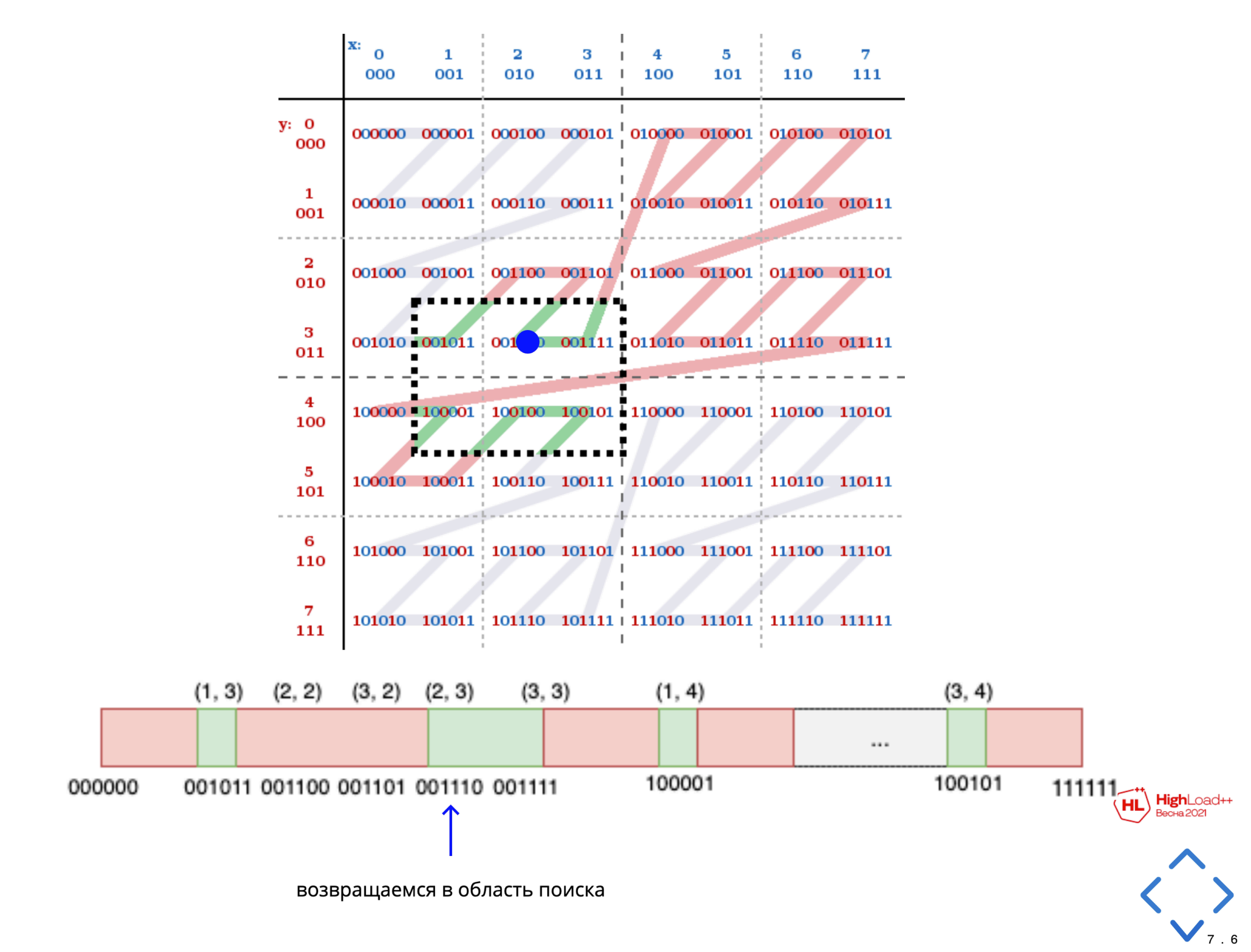




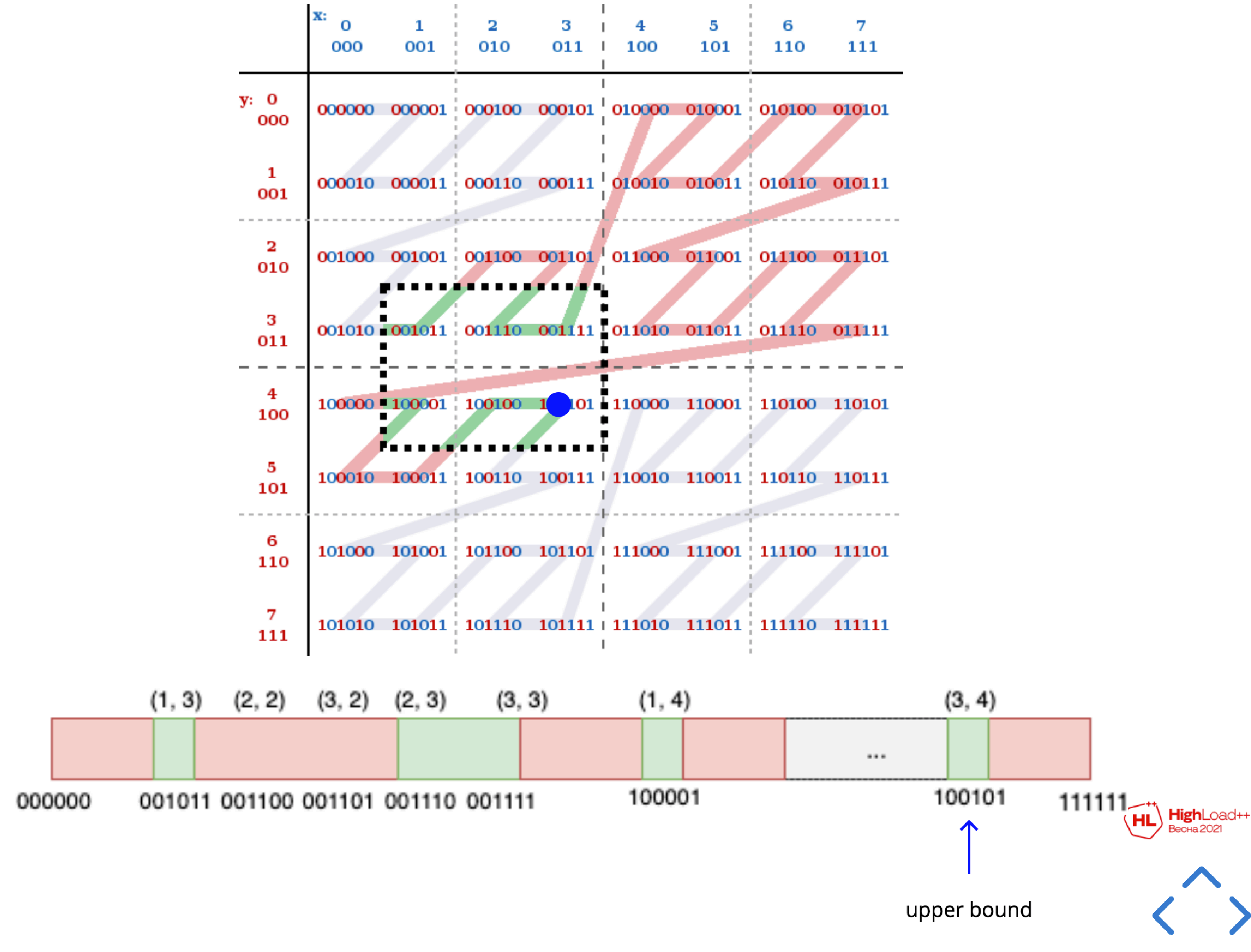
x:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
y: 0 000
000000 000001 000100 000101 010000 010001 010100 010101
1 001
000010 000011 000110 000111 010010 010011 010110 010111
2 010
001000 001001 001100 001101 011000 011001 011100 011101
3 011
001010 001110 001111 011010 011011 011110 011111
4 100
100000 100001 100100 100101 110000 110001 110100 110101
5 101
100010 100011 100110 100111 110010 110011 110110 110111
6 110
101000 101001 101100 101101 111000 111001 111100 111101
7 111
101010 101011 101110 101111 111010 111011 111110 111111
(1, 3) (2, 2) (3, 2) (2, 3) (3, 3) (1, 4) (3, 4)
...
000000 001011 001100 001101 001110 001111 100001 100101 111111
lower bound
HL HighLoad++ Весна 2021



x: 0 000 | 1 001 | 2 010 | 3 011 | 4 100 | 5 101 | 6 110 | 7 111
y: 0 000
000000 000001 000100 000101 010000 010001 010100 010101
1 001
000010 000011 000110 000111 010010 010011 010110 010111
2 010
001000 001001 001101 011000 011001 011100 011101
3 011
001010 001011 001110 001111 011010 011011 011110 011111
4 100
100000 100001 100100 100101 110000 110001 110100 110101
5 101
100010 100011 100110 100111 110010 110011 110110 110111
6 110
101000 101001 101100 101101 111000 111001 111100 111101
7 111
101010 101011 101110 101111 111010 111011 111110 111111
(1, 3) (2, 2) (3, 2) (2, 3) (3, 3) (1, 4) (3, 4)
...
000000 001011 001100 001101 001110 001111 100001 100101 111111
выход за границу
HL HighLoad++ Весна 2021



x:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
y: 0 000
000000 000001 000100 000101 010000 010001 010100 010101
1 001
000010 000011 000110 000111 010010 010011 010110 010111
2 010
001000 001001 001100 001101 011000 011001 011100 011101
3 011
001010 001011 001 001111 011010 011011 011110 011111
4 100
100000 100001 100100 100101 110000 110001 110100 110101
5 101
100010 100011 100110 100111 110010 110011 110110 110111
6 110
101000 101001 101100 101101 111000 111001 111100 111101
7 111
101010 101011 101110 101111 111010 111011 111110 111111
(1, 3) (2, 2) (3, 2) (2, 3) (3, 3) (1, 4) (3, 4)
...
000000 001011 001100 001101 001110 001111 100001 100101 111111
возвращаемся в область поиска
HL HighLoad++ Весна 2021



x:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
y:
0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
000000 000001 000100 000101 010000 010001 010100 010101
000010 000011 000110 000111 010010 010011 010110 010111
001000 001001 001100 001101 011000 011001 011100 011101
001010 001011 001110 001111 011010 011011 011110 011111
100000 100001 100100 110000 110001 110100 110101
100010 100011 100110 100111 110010 110011 110110 110111
101000 101001 101100 101101 111000 111001 111100 111101
101010 101011 101110 101111 111010 111011 111110 111111
(1, 3)
(2, 2)
(3, 2)
(2, 3)
(3, 3)
(1, 4)
(3, 4)
...
000000
001011 001100 001101 001110 001111
100001
100101
111111
upper bound
HL HighLoad++ Весна 2021

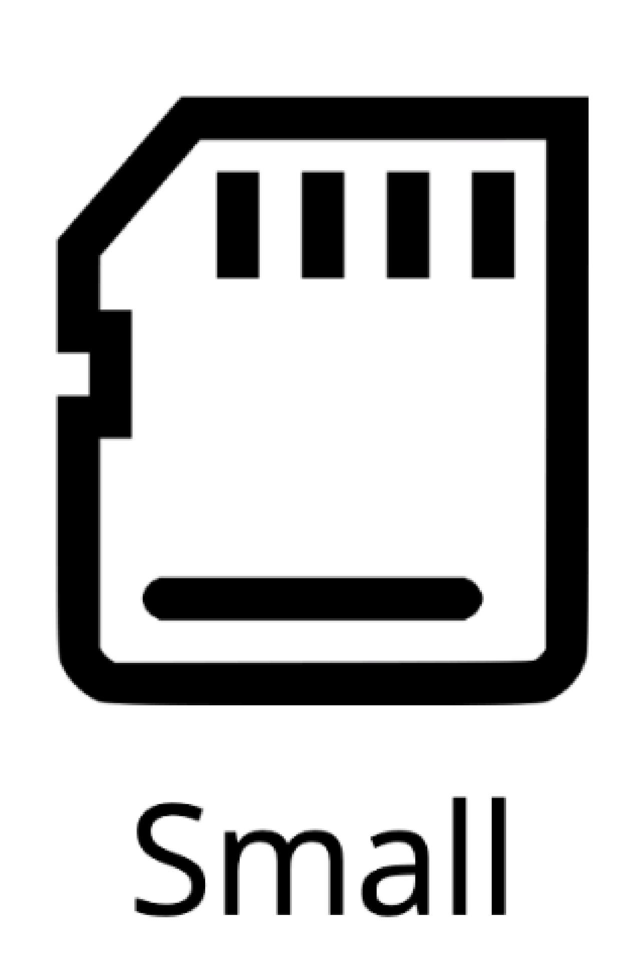
Встраивание в Tarantool

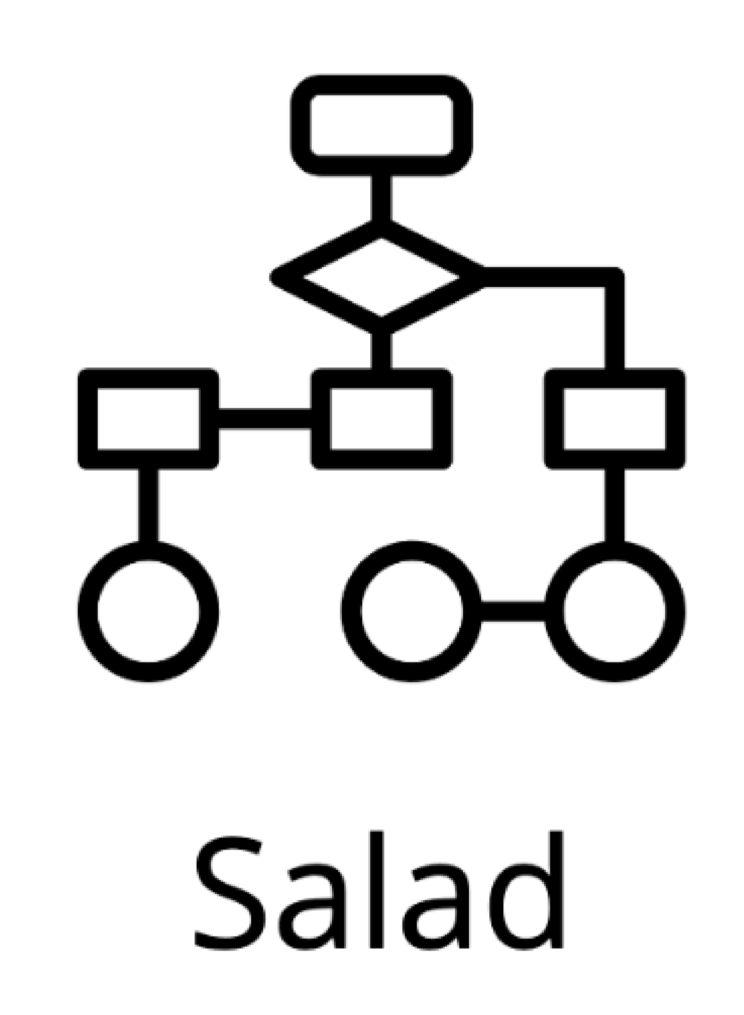
- Вычисление z-адреса
- Сохранение в B+*-Tree
- Реализация алгоритмов поиска

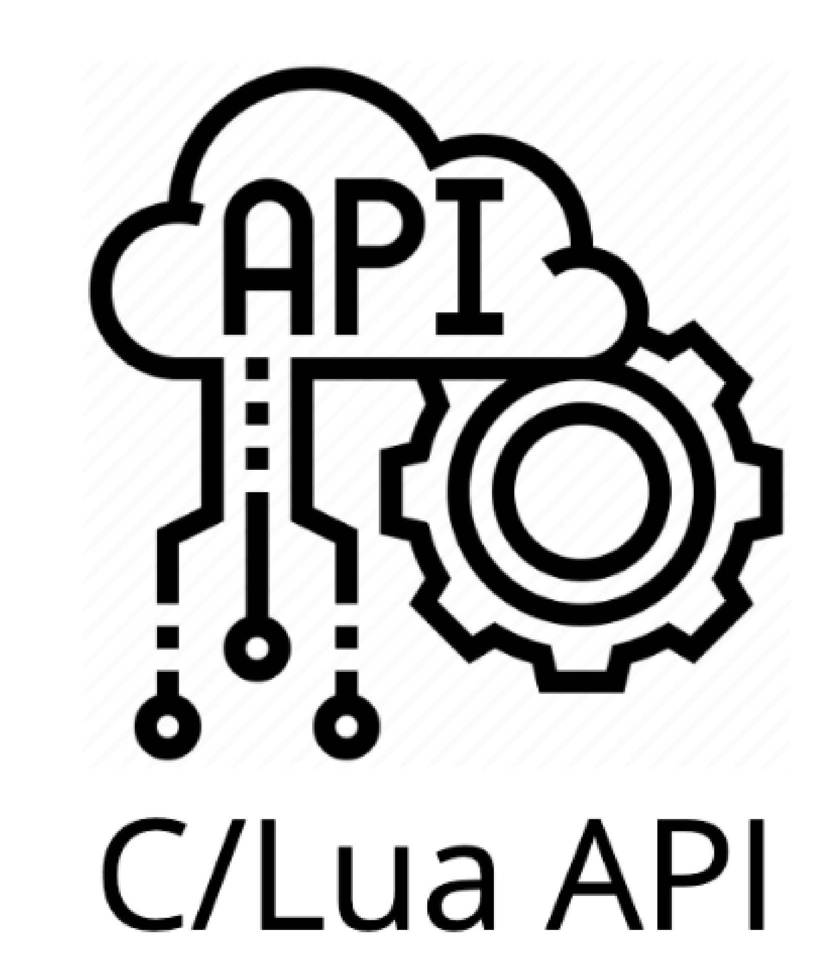




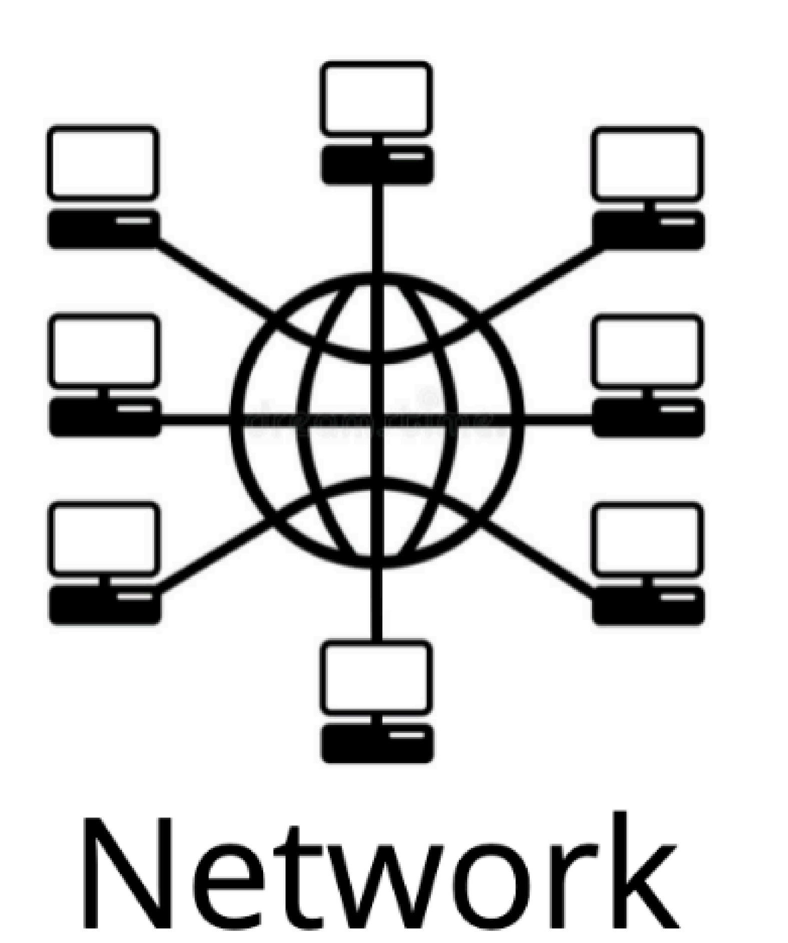

Что доступно разработчику?


Small

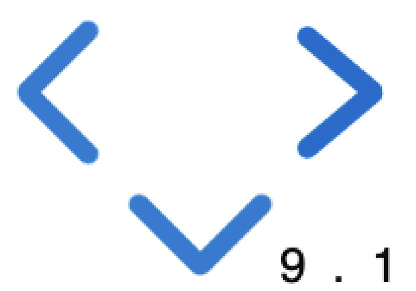

Salad



C/Lua API


Network





Specialized Memory ALLocators

mmap

static buffer

slab arena

static

slab cache

region

lsregion

mempool

ibuf

obuf

matras

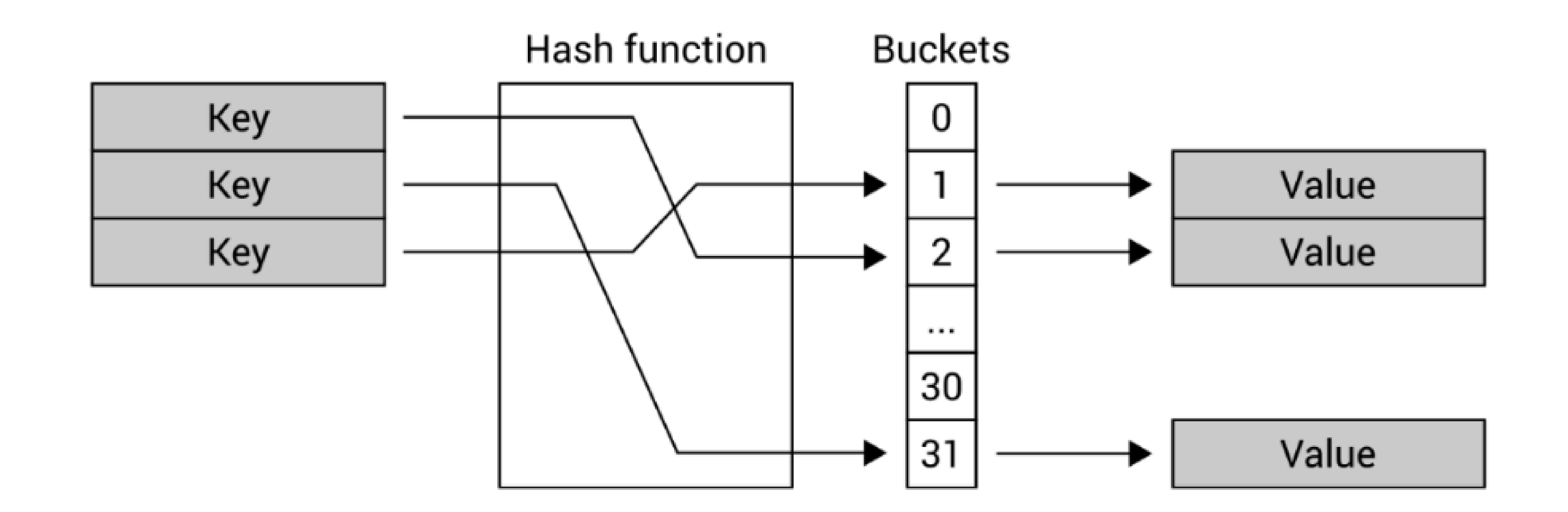
small alloc

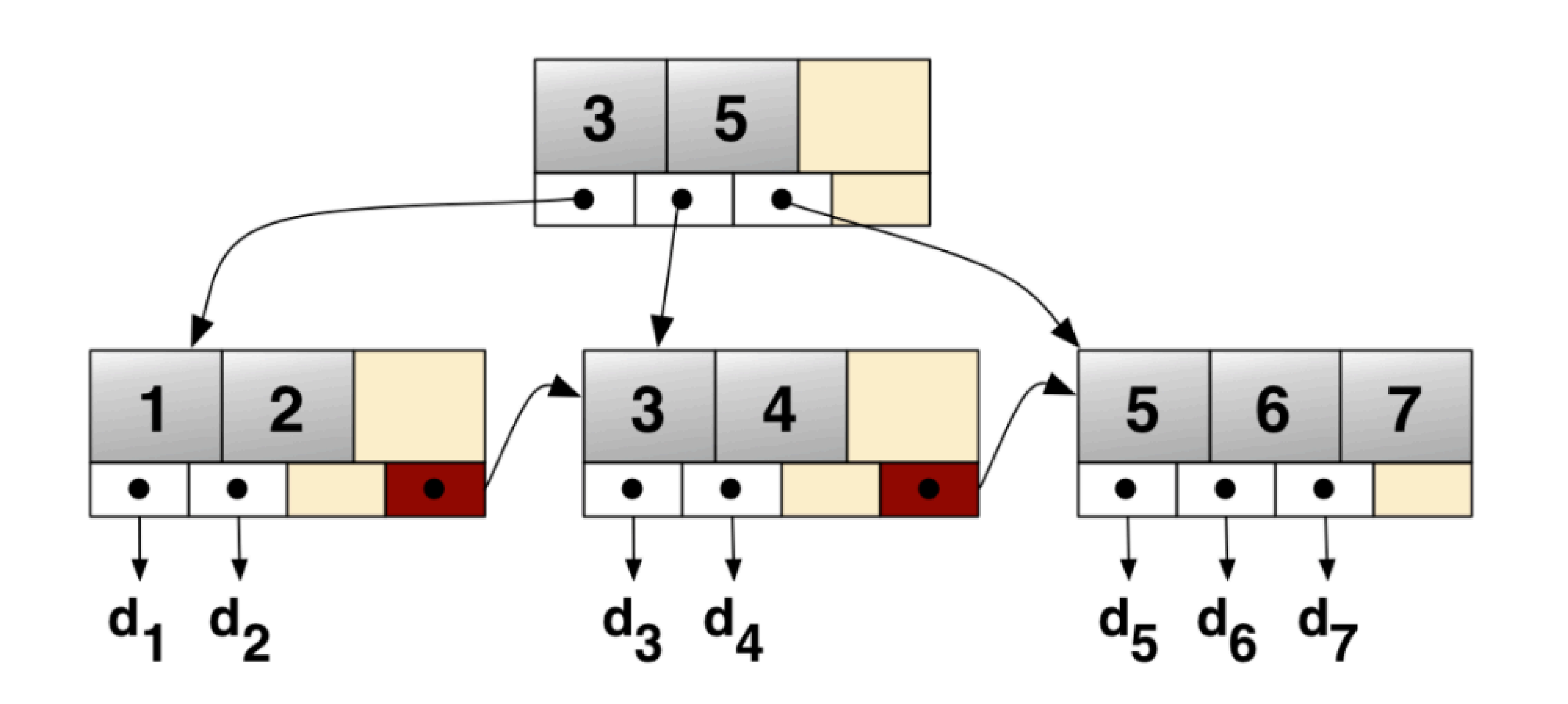
common
network
memtx
vinyl

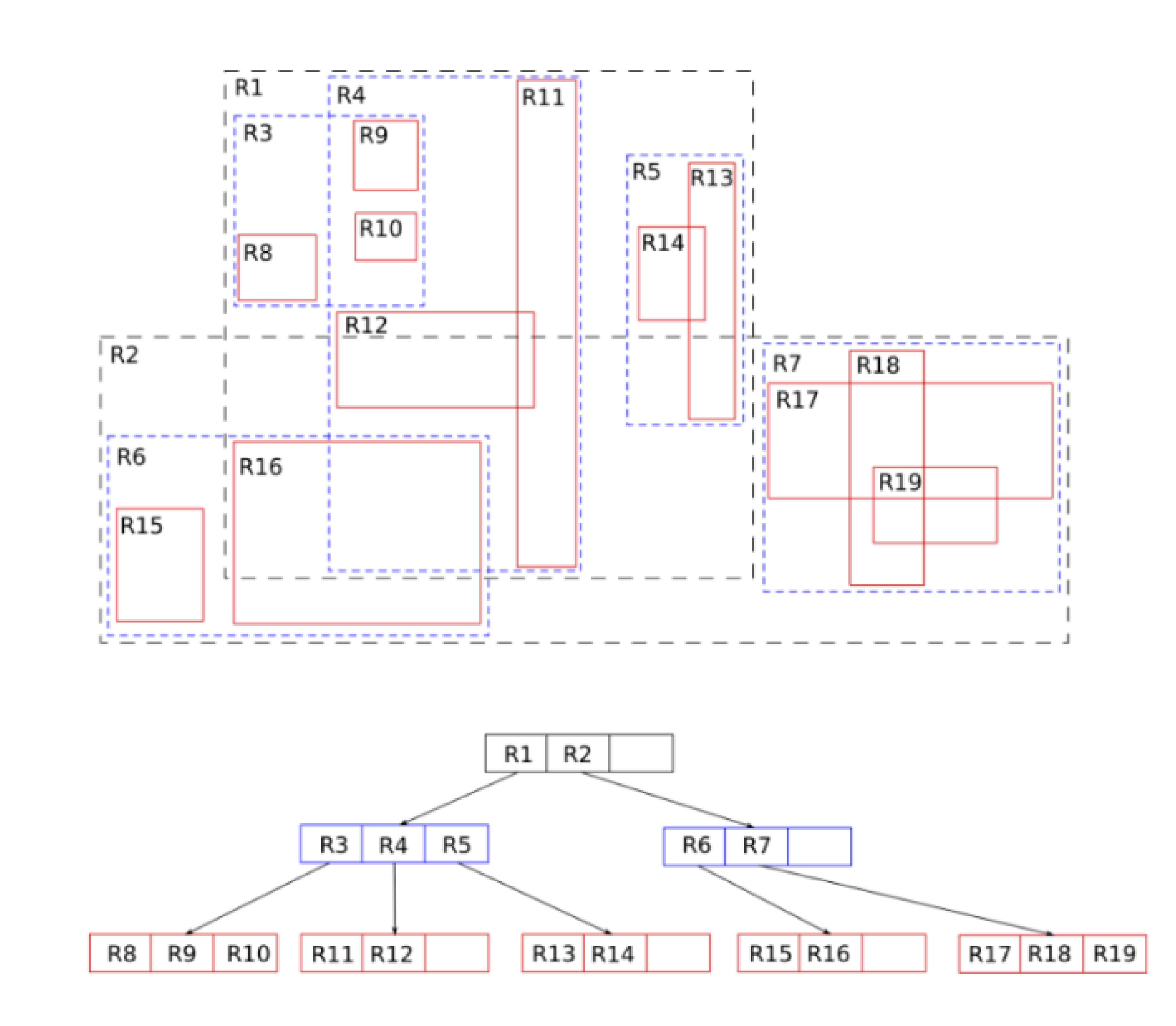




Some ALgorithms And Data structures











C API/Lua API

- Lua C API
- LuaJIT FFI
- Tarantool C API

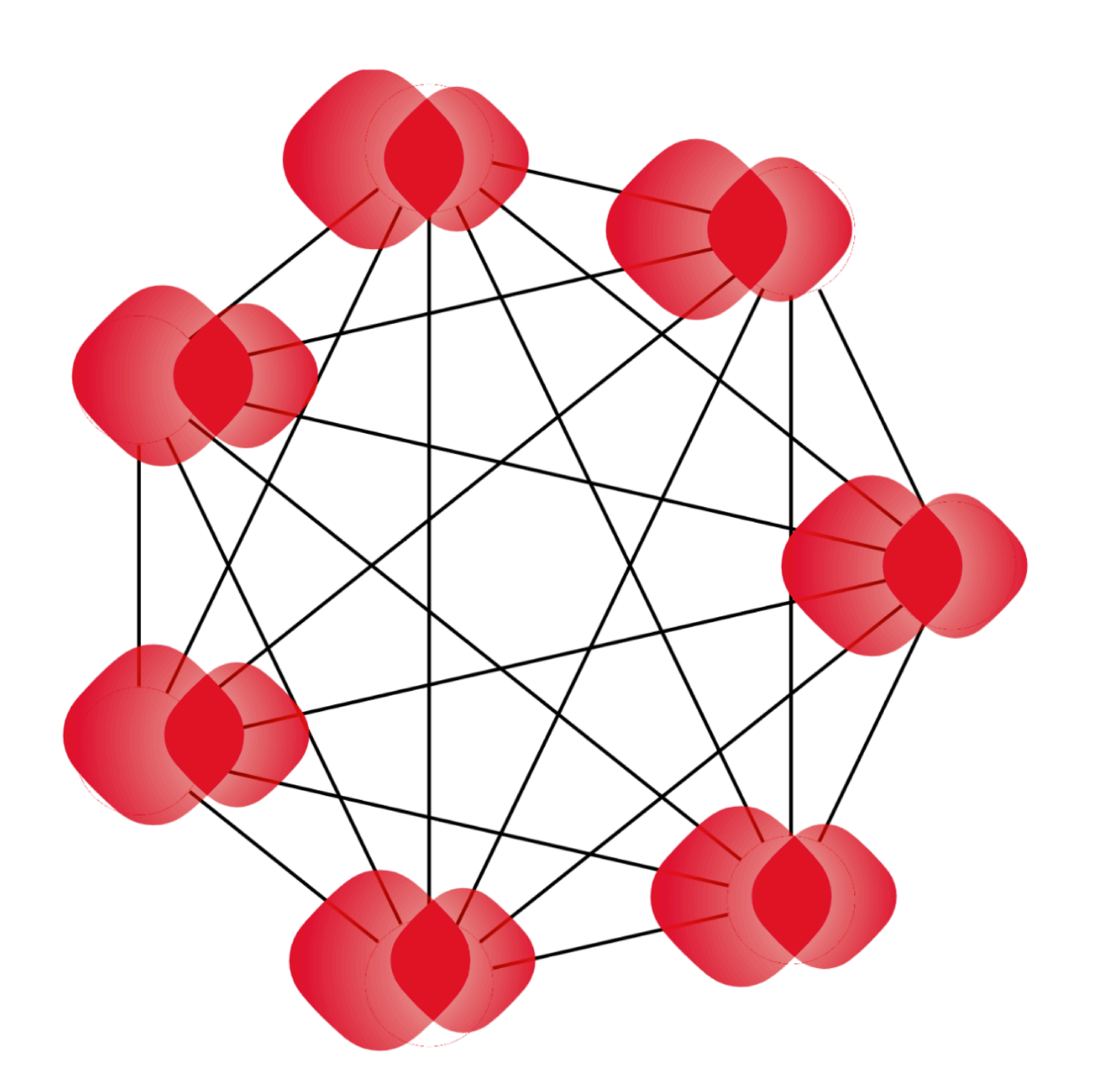




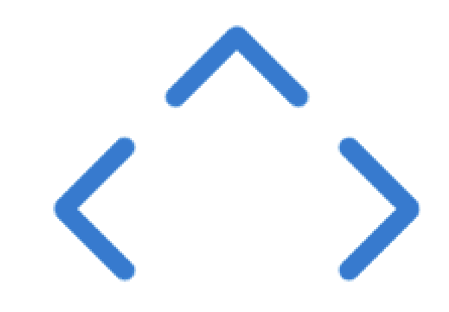


Network

- IProto
  - Call
  - Replication
  - DML
  - RAFT
- SWIM
- ...







Что понадобилось?

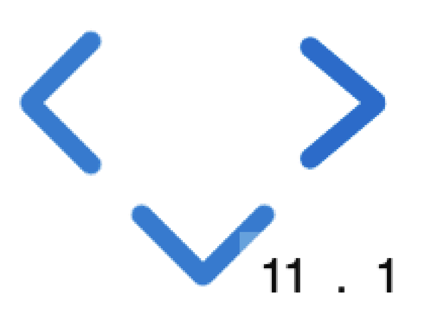
- Набор алгоритмов для работы с Z-адресами
- B+*-Tree
- Bit array





Работа с Z-адресами

HighLoad++
Весна 2021



Вычисление

header

Tuple

field 1 *field2* *field 3*

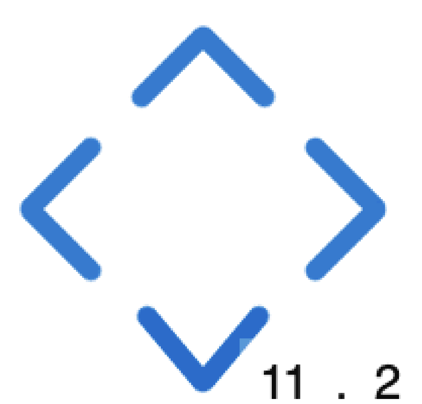
decode & extract

Key 1111 0000

bit interleaving

Z-address 1 0 1 0 1 0 1 0





Хранение в BPS-Tree

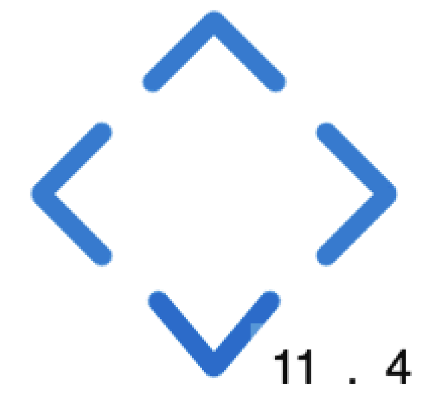
```
struct memtx_tree_data {
        /* Tuple that this node is represents. */
        struct tuple *tuple;
        /** Comparison hint, see key_hint(). */
        hint_t hint;
};
```

```
struct memtx_zcurve_data {
        /* Z-address. Read here: https://en.wikipedia.org/wiki/Z-order_curve */
        z_address *z_address;
        /** Tuple that this node is represents. */
        struct tuple *tuple;
};
```

Работа с типами

Type	Dec	Bin	Normalized
unsigned	6	00000110	00000110
integer	6	00000110	10000110
	-6	10000110	00000110
string	abc	0x61,0x62,0x63	0x61,0x62
	a	0x61	0x61,0x00





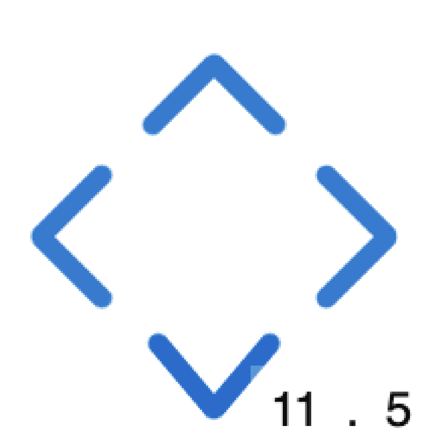
Next-jump-in

y \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11

upper bound - 50



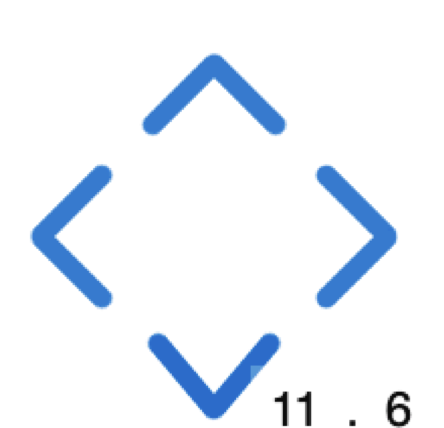


y: \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11

upper bound - 50





y \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

lower bound - 11

upper bound - 50

current - 40





y \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

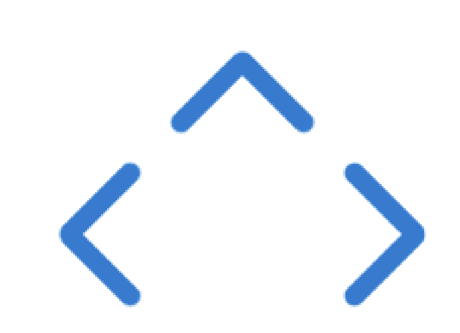
lower bound - 11

upper bound - 50

current - 40

NIP - 48

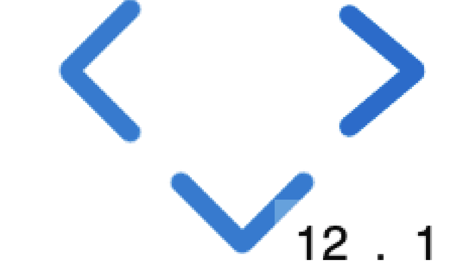




Индекс в Tarantool

- replace
- create_iterator
- get
- count
- size
- bsize
- min
- max
- update_def
- depends_on_pk
- ...

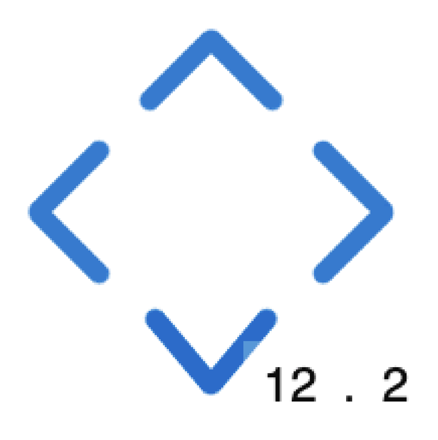




Replace

```
int
memtx_zcurve_index_replace(struct index *base, struct tuple *old_tuple,
                           struct tuple *new_tuple, enum dup_replace_mode mode,
                           struct tuple **result)
```

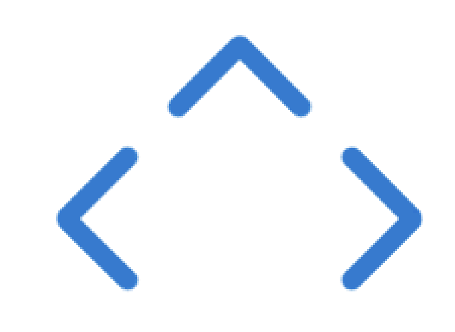




Create iterator

```
struct iterator *
memtx_zcurve_index_create_iterator(struct index *base,
        enum iterator_type type, const char *key, uint32_t part_count)
```

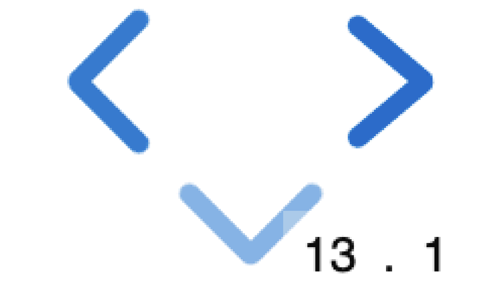




Lua API

```
space = box.schema.space.create('my_space')
space:create_index('primary',
    {type = 'tree', parts = {{1, 'unsigned'}}})
space:create_index('secondary',
    {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
for i = 0, 5 do
    for j = 0, 5 do
        space:insert({i * 6 + j, i, j})
    end
end
```

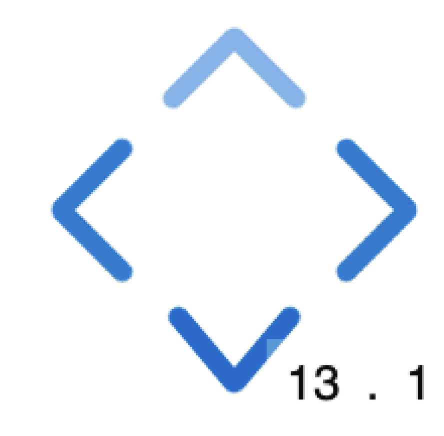




Lua API

```lua
space = box.schema.space.create('my_space')
space:create_index('primary',
    {type = 'tree', parts = {{1, 'unsigned'}}})
space:create_index('secondary',
    {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
for i = 0, 5 do
    for j = 0, 5 do
        space:insert({i * 6 + j, i, j})
    end
end
```

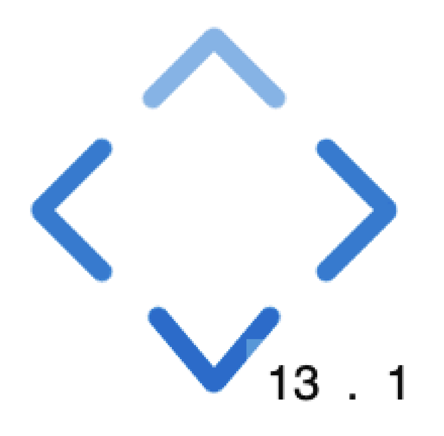




Lua API

```lua
space = box.schema.space.create('my_space')
space:create_index('primary',
    {type = 'tree', parts = {{1, 'unsigned'}}})
space:create_index('secondary',
    {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
for i = 0, 5 do
    for j = 0, 5 do
        space:insert({i * 6 + j, i, j})
    end
end
```

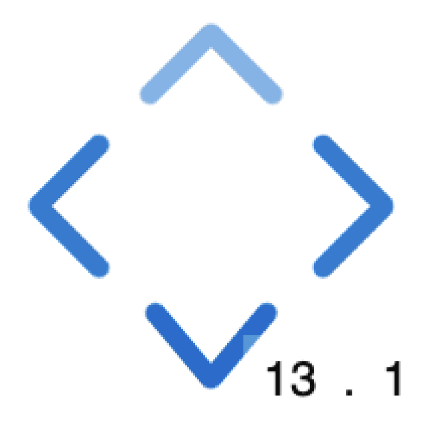




Lua API

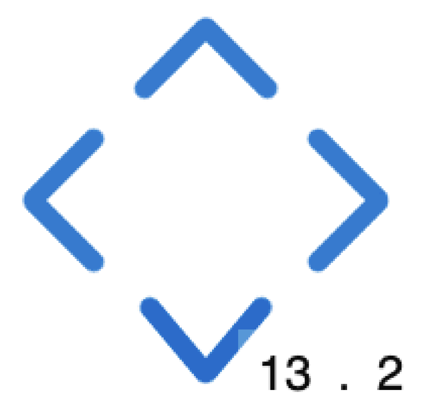
```lua
space = box.schema.space.create('my_space')
space:create_index('primary',
    {type = 'tree', parts = {{1, 'unsigned'}}})
space:create_index('secondary',
    {type = 'zcurve', parts = {{2, 'unsigned'}, {3, 'unsigned'}}})
for i = 0, 5 do
    for j = 0, 5 do
        space:insert({i * 6 + j, i, j})
    end
end
```





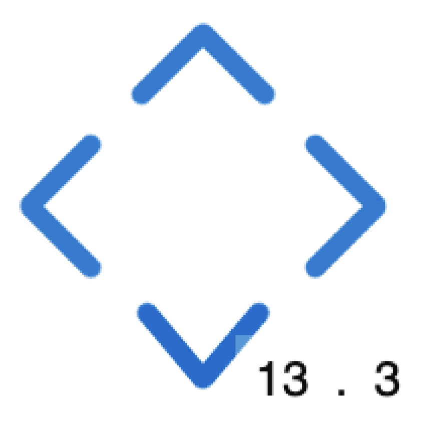
```
-- (2 <= x <= 3) and (3 <= y <= 5)
tarantool> secondary:select({2, 3, 3, 5})
---
- - [15, 2, 3]
  - [21, 3, 3]
  - [16, 2, 4]
  - [22, 3, 4]
  - [17, 2, 5]
  - [23, 3, 5]
...
```





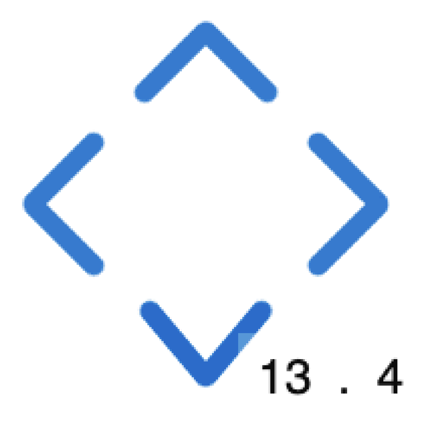
```
-- (x == 2) and (y == 3)
tarantool> secondary:select({2, 3})
---
- - [15, 2, 3]
...
```





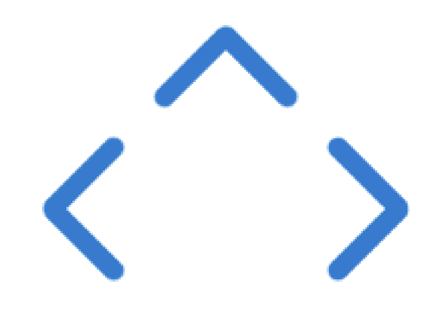
```
-- (2 <= x <= 3)
tarantool> secondary:select({2, 3, box.NULL, box.NULL})
---
- - [12, 2, 0]
  - [18, 3, 0]
  - [13, 2, 1]
  - [19, 3, 1]
  - [14, 2, 2]
  - [20, 3, 2]
  - [15, 2, 3]
  - [21, 3, 3]
  - [16, 2, 4]
  - [22, 3, 4]
  - [17, 2, 5]
  - [23, 3, 5]
...
```





```
-- (x >= 2) and (y >= 3)
tarantool> secondary:select({2, box.NULL, 3, box.NULL})
---
- - [15, 2, 3]
  - [21, 3, 3]
  - [27, 4, 3]
  - [33, 5, 3]
  - [16, 2, 4]
  - [22, 3, 4]
  - [17, 2, 5]
  - [23, 3, 5]
  - [28, 4, 4]
  - [34, 5, 4]
  - [29, 4, 5]
  - [35, 5, 5]
...
```

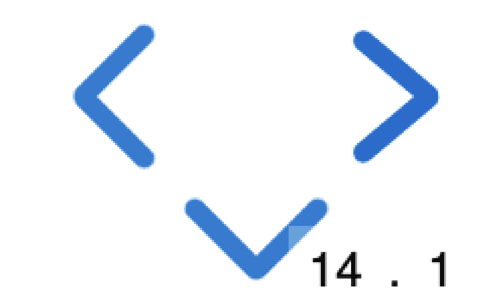




Z-order curve vs B-Tree

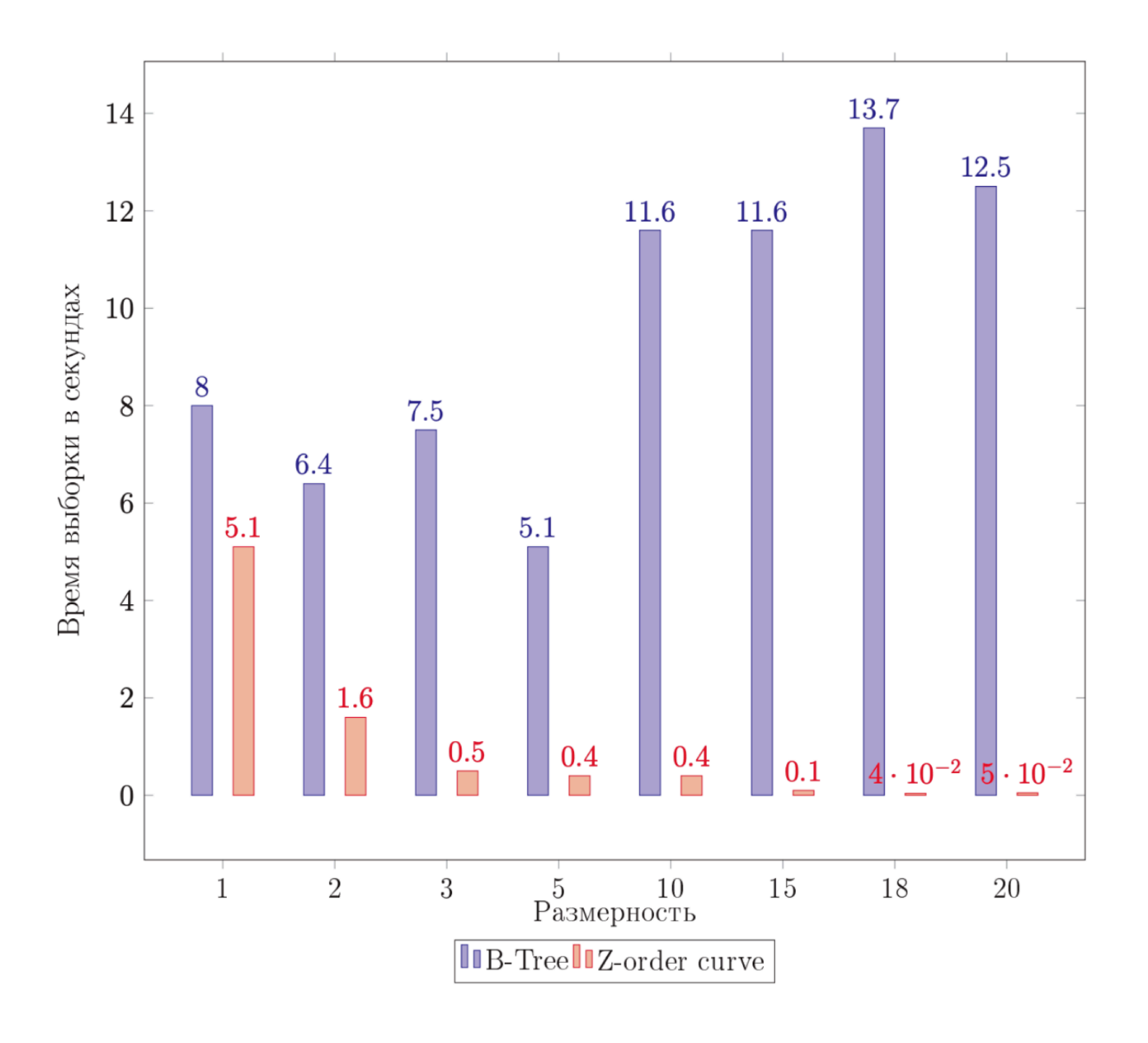
- Поиск в прямоугольной области
- Префиксный поиск





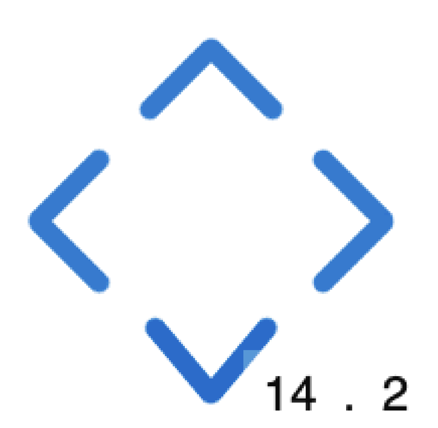
Поиск в прямоугольной области

10^6 точек, распределенных равномерно



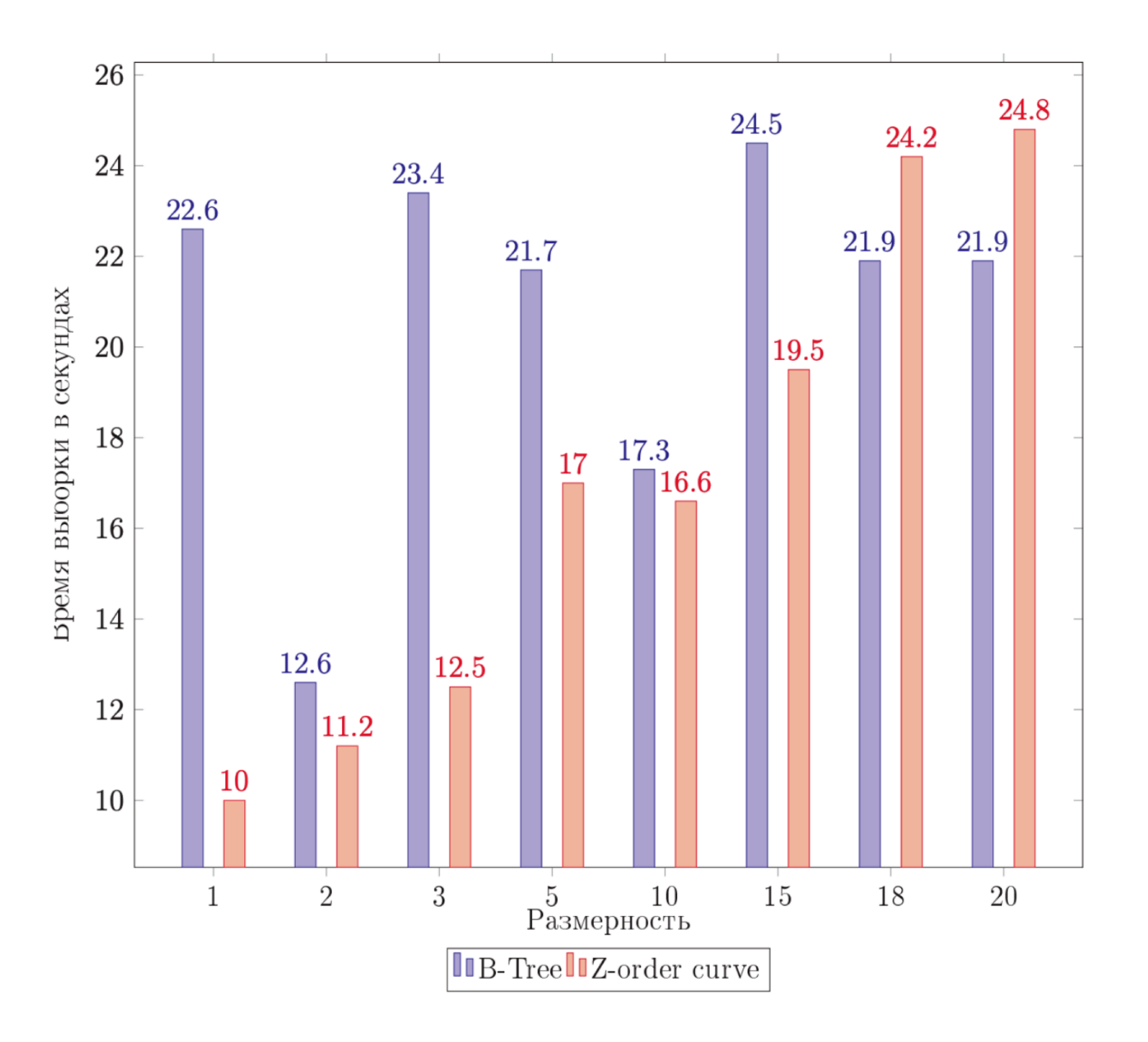
Размерность	B-Tree	Z-order curve
1	8	5.1
2	6.4	1.6
3	7.5	0.5
5	5.1	0.4
10	11.6	0.4
15	11.6	0.1
18	13.7	$4 \cdot 10^{-2}$
20	12.5	$5 \cdot 10^{-2}$



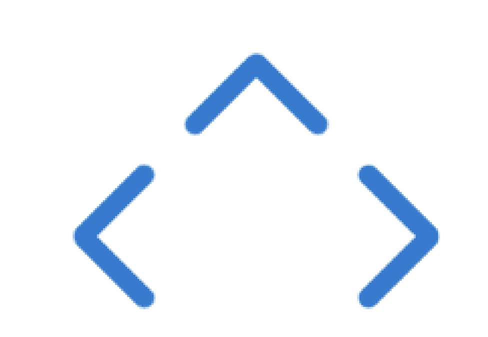


Поиск в прямоугольной области

10^6 точек, смещенных к 0



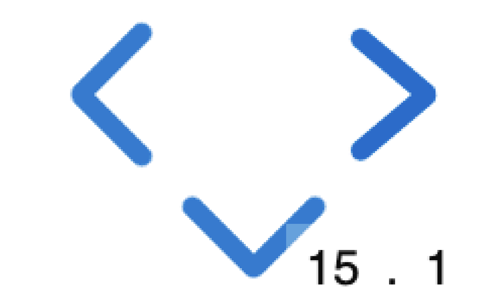




Z-order curve vs R-Tree

- Запись
- Потребляемая память
- Поиск в прямоугольной области



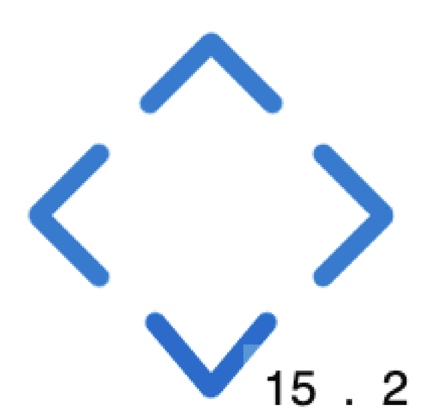


Запись

Время вставки в секундах

Размерность	R-Tree	Z-order curve
1	1.95	2.19
2	2.43	2.53
3	3.21	2.82
5	3.96	3.46
10	6.2	4.41
15	8.84	4.94
18	9.91	5.41
20	10.76	5.71





Потребление памяти

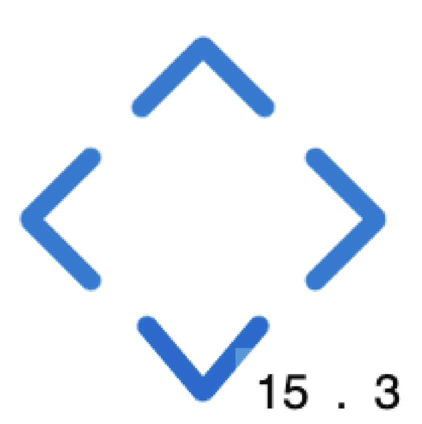
Размерность	R-Tree	Z-order curve
1	36	34
2	60	41
3	85	49
5	129	64
10	245	102
15	352	141
18	425	163
20	475	179

Потребляемая память в мегабайтах

Размерность

R-Tree Z-order curve



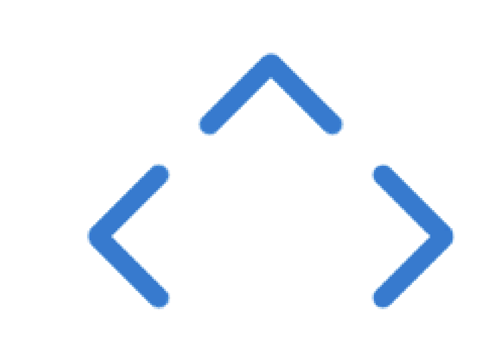


Поиск в прямоугольной области

Время выборки в секундах

Размерность	R-Tree	Z-order curve
1	358.5	383.7
2	204.3	217.4
3	131	160.1
5	46.6	107.7
10	18.7	216.2
15	21.7	531.8
18	21.7	847.1
20	22.9	1,156.9





Почему за год всё сломалось?

 alexlyapunov 8 февраля 2021 в 16:45

Менеджер транзакций для базы данных в оперативной памяти

Блог компании Mail.ru Group, Высокая производительность, Алгоритмы, Хранение данных, Tarantool





Вывод

- Отрицательный результат — тоже результат :)
- Tarantool — отличная платформа для изучения БД
- Не бойтесь экспериментировать





Вопросы

Олег Бабин

tg: @olegrok

e-mail: o.babin@corp.mail.ru

Fork:
https://github.com/olegrok/tarantool/tree/z-order-curve-index



